The Aggregating Algorithm and Regression

Steven Busuttil

Computer Learning Research Centre and
Department of Computer Science,
Royal Holloway, University of London,
United Kingdom

2008

A dissertation submitted in fulfilment of the degree of Doctor of Philosophy.

Declaration

I declare that this dissertation was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Steven Busuttil

Supervisors and Examiners

Supervisors:	*Dr Yuri Kalnishkan* and *Prof Alex Gammerman*
Internal Examiner:	*TBA*
External Examiner:	*TBA*

Abstract

Our main interest is in the problem of making predictions in the online mode of learning where at every step in time a signal arrives and a prediction needs to be made before the corresponding outcome arrives. Loss is suffered if the prediction and outcome do not match perfectly. In the prediction with expert advice framework, this protocol is augmented by a pool of experts that produce their predictions before we have to make ours. The Aggregating Algorithm (AA) is a technique that optimally merges these experts so that the resulting strategy suffers a cumulative loss that is almost as good as that of the best expert in the pool.

The AA was applied to the problem of regression, where outcomes are continuous real numbers, to get the AA for Regression (AAR) and its kernel version, KAAR. On typical datasets, KAAR's empirical performance is not as good as that of Kernel Ridge Regression (KRR) which is a popular regression method. KAAR performs better than KRR only when the data is corrupted with lots of noise or contains severe outliers. To alleviate this we introduce methods that are a hybrid between KRR and KAAR. Empirical experiments suggest that, in general, these new methods perform as good as or better than both KRR and KAAR.

In the second part of this dissertation we deal with a more difficult problem — we allow the dependence of outcomes on signals to change with time. To handle this we propose two new methods: WeCKAAR and KAARCh. WeCKAAR is a simple modification of one of our methods from the first part of the dissertation to include decaying weights. KAARCh is an application of the AA to the case where the experts are all the predictors that can change with time. We show that KAARCh suffers a cumulative loss that is almost as good as that of any expert that does not change very rapidly. Empirical results on data with changing dependencies demonstrate that WeCKAAR and KAARCh perform well in practice and are considerably better than Kernel Ridge Regression.

To Mariella

Acknowledgements

I am grateful to my supervisor Yuri Kalnishkan for providing help, direction and insight during my PhD. I also thank my other supervisor Alex Gammerman for helpful discussions and comments. I am indebted to Volodya Vovk who was always happy to provide advice and ideas. I also thank Zhiyuan Luo for interesting and helpful discussions.

During my PhD I made several friends at the department, in particular Tony Bellotti, Brian Burford, Mikhail Dashevsky, and Joe Reddington. I thank them for discussions we had, both related and unrelated to my PhD, the occasional walk or visit to the pub, and for their company during a break. I am thankful to Dr Michael Vyugin at the RTSSE for suggesting a problem which inspired part of this work, for providing data, and for sharing his expertise with us.

I also thank all the members of the Computer Centre Badminton club for helping me in my attempts to keep fit and in learning the sport. I am also grateful to the support and administrative teams for their help and for keeping the department running smoothly.

I thank my parents, brothers, sisters and the Fardell family for their continuous love and support. Above all, I thank my wife Mariella, for her love, support and patience.

This research was made possible by the Thomas Holloway studentship I was awarded from Royal Holloway College, for which I am very grateful. I also thank the college for funding my trips abroad to present papers at conferences.

Contents

List of Figures

List of Tables

Lists of Theorems, Corollaries and Lemmas

Notation

$\mathbb{N}$	The space of natural numbers.
$\mathbb{R}$	The space of real numbers.
$\mathbb{R}^n$	The n-dimensional space of real numbers.
$\mathcal{H}$	A Hilbert space $\mathcal{H}$; uppercase script letters are used for Hilbert spaces.
$\mathbf{B}$	A matrix $\mathbf{B}$; bold uppercase letters are used for matrices.
$\mathbf{B}'$	Transpose of matrix $\mathbf{B}$.
$\mathbf{v} = (v_1, \ldots, v_n)'$	The n-dimensional (column) vector $\mathbf{v}$ with elements $v_1, \ldots, v_n$; bold lowercase letters are used for vectors.
$\langle \mathbf{x}, \mathbf{z} \rangle$	The dot product of vectors $\mathbf{x}$ and $\mathbf{z}$.
$\mathrm{Loss}_T(S)$	Loss of S at time T.
$\mathcal{L}_T(S)$	Objective loss function of S at time T.

Acronyms

AA	Aggregating Algorithm
AAR	Aggregating Algorithm for Regression
AARCh	Aggregating Algorithm for Regression with Changing dependencies
ANOVA	ANalysis Of VAriance
APA	Aggregating Pseudo Algorithm
CKAAR	Controlled KAAR
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
IKAAR	Iterated KAAR
KAAR	Kernel Aggregating Algorithm for Regression
KAARCh	Kernel Aggregating Algorithm for Regression with Changing dependencies
KOKO	Convex combination of KRR and KAAR
KRR	Kernel Ridge Regression
KRRV	Scaled version of KRR
LS	Least Squares
RBF	Radial Basis Function
RKHS	Reproducing Kernel Hilbert Space
RR	Ridge Regression
RTSSE	Russian Trading System Stock Exchange
WeCKAAR	Weighted CKAAR

Chapter 1

Introduction

- Is it going to rain tomorrow?
- How much is my house worth?
- Given a new protein sequence, what is its type?
- What is the price of a stock market share going to be in an hour's time?

It would be desirable if we could answer questions like the ones above quickly, cheaply and accurately. However, this may not be possible if we use traditional methods. For instance, implementing an exact solution may be too complex or, as for the case of protein classification, the traditional (laboratory) techniques can be expensive and time consuming. And yet, for some, we simply do not know of an exact solution.

This is were machine learning comes in. Since, for some reason or other, we cannot have an exact solution to a problem, we write a computer program that can *learn* a good solution instead. The definition of machine learning as given in Mitchell [1997, Section 1.1] is

Definition 1 (Machine Learning) A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. □

When the class of predictions is discrete (for example, yes/no answers), the problem is known as that of classification. On the other hand, if the answers are continuous, for example, when we have to predict the price of a share, the problem is known as regression. Learning can be made in batch mode or online mode. In the batch mode of learning we have a fixed training set which our algorithm can learn from and testing examples to make predictions on. In online mode, examples arrive one by one and at each step we have to make a prediction for each one. In this dissertation we are mostly interested in making predictions for regression in the online mode of learning.

In the online protocol, on each trial (or step) $t = 1, 2, \ldots$ the learner observes a signal $\mathbf{x}_t$ and attempts to predict the outcome y_t, which is shown to the learner later. At each step the learner suffers loss. There are different types of losses; however, we are mostly interested in the square loss which is the squared discrepancy between a prediction and the outcome. The overall performance of the learner is measured by means of the sum of all these losses, known as the cumulative loss. A popular solution to the regression problem is Ridge Regression (RR), introduced to statistics in Hoerl [1962]. This aims to find a solution that is simple and that minimises the square loss suffered on a training set. A nonlinear version of RR, known as Kernel Ridge Regression (KRR), was subsequently derived through the use of kernels (see Saunders et al. [1998]).

The Aggregating Algorithm (AA), introduced in Vovk [1990, 1998], allows us to merge experts from large pools to obtain optimal strategies. Such an optimal strategy performs nearly as good as the best expert from the class in terms of the cumulative loss it suffers. In Vovk [2001] the AA is applied to merge all constant linear predictors, i.e., experts θ predicting $\theta' \mathbf{x}_t$ (it is assumed that $\mathbf{x}_t$ and θ are drawn from $\mathbb{R}^n$). The resulting Aggregating Algorithm for Regression (AAR) (also known as the Vovk-Azoury-Warmuth forecaster, see Cesa-Bianchi and Lugosi [2006, Section 11.8]) performs almost as well as the best predictor θ. In Gammerman et al. [2004] the kernel version of AAR, known as the Kernel AAR (KAAR), is introduced and a bound on its performance is derived (see also Vovk [2006, Section 8]). From a computational point of view the algorithm is similar to Ridge Regression.

1.1 Research Objectives

The objectives of this dissertation are *to analyse existing applications of the Aggregating Algorithm (AA) to the problem of regression, to improve them* and *to create new methods for regression based on the AA.*

1.2 Original Contributions

In the first part of this dissertation we analyse the empirical performance of KAAR. We notice that although KAAR has better theoretical properties than KRR, the latter tends to perform much better in most cases. KAAR is better than KRR only when the data contains severe outliers or is corrupted with lots of noise. We therefore suggest several improvements to KAAR. This results in mainly two new hybrid algorithms that have an extra parameter with which they can be made to behave like KRR or KAAR. We then proceed to give a new Bayesian interpretation of KAAR and our methods. Empirical experiments suggest that, in general, these new methods suffer a loss that is less or equal to that of KRR and KAAR.

In the second part of this dissertation we deal with a more difficult regression problem. Usually, it is assumed that the dependency of y_t on $\mathbf{x}_t$ is fixed. We are interested in the case where this dependency can change with time. An example of where this might be applicable is in the prediction of financial option implied volatility which is known to exhibit a dependence on time. Standard regression techniques like KRR do not handle this problem well. We therefore introduce two new methods: WeCKAAR and KAARCh.

WeCKAAR is a simple method that adds decaying weights to one of our hybrid regression techniques. KAARCh is a new method based on the Aggregating Algorithm (AA). To get KAARCh, the AA is used to merge all predictors that can change with time. We show that KAARCh performs almost as well as any predictor if the latter is not changing very rapidly. It turns out that both WeCKAAR and KAARCh are, once again, computationally similar to KRR. Empirical experiments on artificial and options implied volatility data suggest that these methods perform well in practice.

1.2.1 Summary of Original Contributions

1. Empirical analysis of KAAR.

2. Derivation of new methods that improve KAAR's empirical performance and are competitive with Kernel Ridge Regression.

3. Bayesian interpretation of KAAR and our new methods where we show that these methods push KRR's prediction towards the mean with an amount proportional to the variance of the prediction itself.

4. Empirical experiments on the methods above.

5. Derivation of WeCKAAR, which is a simple modification of one of the techniques mentioned in item 2.

6. Derivation of KAARCh by applying the AA to a class of predictors that can change with time.

7. Theoretical upper bound that shows that KAARCh's loss is less or equal to that of any predictor that does not change very rapidly, plus a small term.

8. Empirical experiments on WeCKAAR and KAARCh.

1.3 List Of Publications

For items 1 to 4 in Section 1.2.1, corresponding to material in Chapter 4:

- S. Busuttil, Y. Kalnishkan, A. Gammerman, and V. Vovk. The kernel aggregating algorithm for regression. *Machine Learning*, submitted.

- S. Busuttil, Y. Kalnishkan, and A. Gammerman. Improving the aggregating algorithm for regression. In *Proceedings of the 25th IASTED International Conference on Artificial Intelligence and Applications (AIA 2007)*, pages 347–352. ACTA Press, 2007.

- S. Busuttil, Y. Kalnishkan, and A. Gammerman. Two new kernel least squares based methods for regression. Technical Report CLRC-TR-06-01, Royal Holloway, University of London, UK, 2006. URL `http://www.clrc.rhul.ac.uk/publications/files/tr0601.pdf`.

For items 5 to 8 in Section 1.2.1, corresponding to material in Chapter 5:

- S. Busuttil and Y. Kalnishkan. Online regression competitive with changing predictors. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory (ALT 2007)*, Lecture Notes in Artificial Intelligence, pages 181–195. Springer, Germany, 2007.

- S. Busuttil and Y. Kalnishkan. Weighted kernel regression for predicting changing dependencies. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, Lecture Notes in Artificial Intelligence, pages 535–542. Springer, Germany, 2007.

1.4 Organisation of the Dissertation

The Aggregating Algorithm (AA), which is at the heart of our work, is described in detail in Chapter 2, while in Chapter 3 we present the problem we are mainly interested in: that of online regression. The first part of our work, where we introduce KAAR and our improvements, is in Chapter 4, and we deal with the problem of regression with changing dependencies in Chapter 5. Finally, in Chapter 6 we conclude this dissertation with some closing remarks and future work. Appendix A contains lemmas used throughout this dissertation, while Appendix B contains more empirical results to support Chapter 4.

Chapter 2

The Aggregating Algorithm

In this chapter we give an overview of the Aggregating Algorithm (AA) following Vovk [2001, Sections 1 and 2]. Note, however, that the presentation of most of the proofs is ours. Informally, given a pool of experts, the AA makes predictions such that it performs almost as well as the best expert in the pool under any circumstances.

2.1 Preliminaries

Let Ω be an outcome space, Γ be a prediction space and Θ be a (possibly infinite) pool of experts. We consider the following game between Statistician (or Learner) S, Nature, and Θ:

for $t = 1, 2, \ldots$ **do**
  Every expert $\theta \in \Theta$ makes a prediction $\gamma_t^{(\theta)} \in \Gamma$
  Statistician S observes all $\gamma_t^{(\theta)}$
  Statistician S outputs a prediction $\gamma_t \in \Gamma$
  Nature outputs $\omega_t \in \Omega$
end for

Given a fixed loss function $\lambda : \Omega \times \Gamma \mapsto [0, \infty]$, Statistician aims to suffer a cumulative loss

$$\mathrm{Loss}_T(S) = \sum_{t=1}^{T} \lambda(\omega_t, \gamma_t)$$

that is not much larger than the loss

$$\mathrm{Loss}_T(\theta) = \sum_{t=1}^{T} \lambda(\omega_t, \gamma_t^{(\theta)})$$

of the best expert $\theta \in \Theta$.

2.2 Algorithm

The AA takes two parameters, a prior probability distribution P_0 in the pool of experts Θ and a learning rate $\eta > 0$. P_0 specifies the initial weights given to the experts. Let $\beta = e^{-\eta}$.

We will first describe the Aggregating Pseudo Algorithm (APA) that does not output actual predictions but generalised predictions. A generalised prediction $g : \Omega \mapsto \mathbb{R}$ is a mapping giving a value of loss for each possible outcome. At every step t, the APA updates the experts' weights so that those that suffered large loss during the previous step have their weights reduced:

$$P_t(d\theta) = \beta^{\lambda(\omega_t, \gamma_t^{(\theta)})} P_{t-1}(d\theta) \ , \qquad \theta \in \Theta \ . \tag{2.1}$$

At time t, the APA chooses a generalised prediction by

$$g_t(\omega) = \log_\beta \int_\Theta \beta^{\lambda(\omega, \gamma_t^{(\theta)})} P^*_{t-1}(d\theta) \ , \tag{2.2}$$

where $P^*_{t-1}(d\theta)$ are the normalised weights

$$P^*_{t-1}(d\theta) = \frac{P_{t-1}(d\theta)}{P_{t-1}(\Theta)} \ .$$

Lemma 1 (Vovk [2001, Lemma 1]) *For any learning rate $\eta > 0$, prior P_0, and $T = 1, 2, \ldots$*

$$\mathrm{Loss}_T(APA) = \log_\beta \int_\Theta \beta^{\mathrm{Loss}_T(\theta)} P_0(d\theta) \ . \tag{2.3}$$

□

PROOF Equation (2.3) holds for $T = 1$ because by using (2.2) we get

$$\begin{aligned}\mathrm{Loss}_1(\mathrm{APA}) &= g_1(\omega_1) \\ &= \log_\beta \int_\Theta \beta^{\lambda(\omega_1,\gamma_1^{(\theta)})} P_0(d\theta) \\ &= \log_\beta \int_\Theta \beta^{\mathrm{Loss}_1(\theta)} P_0(d\theta) \ .\end{aligned}$$

Let us assume that (2.3) holds at step $T-1$. We will now show that it also holds at step T. Clearly,

$$\begin{aligned}\mathrm{Loss}_T(\mathrm{APA}) &= g_T(\omega_T) + \mathrm{Loss}_{T-1}(\mathrm{APA}) \\ &= \log_\beta \frac{\int_\Theta \beta^{\lambda(\omega_T,\gamma_T^{(\theta)})} P_{T-1}(d\theta)}{P_{T-1}(\Theta)} + \log_\beta \int_\Theta \beta^{\mathrm{Loss}_{T-1}(\theta)} P_0(d\theta) \ .\end{aligned}$$

Notice that from (2.1) it follows that

$$\begin{aligned}P_{T-1}(d\theta) &= \beta^{\lambda(\omega_{T-1},\gamma_{T-1}^{(\theta)})+\cdots+\lambda(\omega_1,\gamma_1^{(\theta)})} P_0(d\theta) \\ &= \beta^{\mathrm{Loss}_{T-1}(\theta)} P_0(d\theta) \ . \end{aligned} \tag{2.4}$$

Therefore,

$$\begin{aligned}\mathrm{Loss}_T(\mathrm{APA}) &= \log_\beta \frac{\int_\Theta \beta^{\lambda(\omega_T,\gamma_T^{(\theta)})+\mathrm{Loss}_{T-1}(\theta)} P_0(d\theta)}{P_{T-1}(\Theta)} \\ &\quad + \ \log_\beta \int_\Theta \frac{\beta^{\mathrm{Loss}_{T-1}(\theta)} P_{T-1}(d\theta)}{\beta^{\mathrm{Loss}_{T-1}(\theta)}} \\ &= \log_\beta \frac{\int_\Theta \beta^{\mathrm{Loss}_T(\theta)} P_0(d\theta) P_{T-1}(\Theta)}{P_{T-1}(\Theta)} \\ &= \log_\beta \int_\Theta \beta^{\mathrm{Loss}_T(\theta)} P_0(d\theta) \ .\end{aligned}$$

Since (2.3) holds for $T = 1$ and also for T if it holds for $T-1$, then it follows by the inductive principle that it holds for any $T \geq 1$. ∎

To get a prediction from the generalised prediction $g_t(\omega)$ (note that we use ω since we do not yet know the real outcome of step t, ω_t) the AA uses a substitution function Σ mapping generalised predictions into Γ. A substitution

function may introduce extra loss; however, in many cases perfect substitution is possible. We say that the loss function λ is η-mixable if there is a substitution function Σ such that

$$\lambda(\omega_t, \Sigma(g_t(\omega))) \leq g_t(\omega_t) \tag{2.5}$$

on every step t, all experts' predictions and all outcomes. The loss function λ is perfectly mixable if it is η-mixable for some $\eta > 0$. In this dissertation we are only interested in the square loss, which is perfectly mixable (see Section 2.3). For information on the case where the loss function is not perfectly mixable see, for example, Vovk [2001, Section 2.1].

Theorem 1 *If the loss function in a game is η-mixable, then the following upper bound on the cumulative loss of the AA holds in this game for any T:*

$$\mathrm{Loss}_T(AA) \leq \log_\beta \int_\Theta \beta^{\mathrm{Loss}_T(\theta)} P_0(d\theta) \ . \qquad \square$$

PROOF This follows immediately from (2.3) and (2.5). ■

In particular, when the pool of experts is finite and all experts are assigned equal prior weights $1/m$, where m is the number of experts, we get, for any $\theta \in \Theta$

$$\begin{aligned} \mathrm{Loss}_T(\mathrm{AA}) &\leq \log_\beta \left(\frac{1}{m} \sum_{\theta \in \Theta} \beta^{\mathrm{Loss}_T(\theta)} \right) \\ &\leq \log_\beta \left(\frac{1}{m} \beta^{\mathrm{Loss}_T(\theta)} \right) \\ &= \mathrm{Loss}_T(\theta) + \frac{\ln m}{\eta} \ . \end{aligned}$$

This bound can be shown to be optimal in a very strong sense for all algorithms attempting to merge experts' predictions (see Vovk [1998]).

2.3 The Square Loss Game

In this dissertation we are concerned with the (bounded) square loss game (see Vovk [2001, Section 2.4]), where $\Omega = [-Y, Y]$, $Y \in \mathbb{R}$, $\Gamma = \mathbb{R}$, and

$\lambda(\omega, \gamma) = (\omega - \gamma)^2$. We need to find the values of η for which this game is η-mixable and a suitable substitution function. In Section 2.3.1 we give these results for the restricted square loss game where $\Omega = \{-1, 1\}$. This restriction is removed in Section 2.3.2.

2.3.1 Results for the Restricted Game

In the restricted square loss game it is required that outcomes $\omega_t \in \{-1, 1\}$.

Lemma 2 (Vovk [2001, Lemma 2]) *The restricted square loss game is η-mixable if and only if $\eta \leq \frac{1}{2}$.* □

Proof The pseudoprediction $(g(-1), g(1))$ in the restricted square loss game can be represented by a point on a plane. Similarly, the set of permitted predictions can be represented by the losses curve

$$\left((-1-\gamma)^2, (1-\gamma)^2\right) \quad ,$$

where $\gamma \in [-1, 1]$, plotted in Figure 2.1.

In the AA, the pseudoprediction $(g(-1), g(1))$ is transformed to the point

$$(x, y) = \left(e^{-\eta g(-1)}, e^{-\eta g(1)}\right) \in [0, 1]^2$$

of the (x, y)-plane. Under this transformation, the set of permitted predictions will be represented by the parametric curve

$$(u, v) = \left(e^{-\eta(-1-\gamma)^2}, e^{-\eta(1-\gamma)^2}\right) \quad ,$$

where, once again, γ ranges over $[-1, 1]$. In Figure 2.2 we plot this curve for 3 different values of η. The game is η-mixable if and only if this curve is convex, since (x, y) would be below the curve in Figure 2.2, which translates to the pseudoprediction $(g(-1), g(1))$ being above the losses curve in Figure 2.1. In this case it would always be possible to find a point on the curve (which corresponds to finding a permitted prediction) that suffers a loss that is less or equal to that of the pseudoprediction. In Figure 2.2 the convex curves are those for which $\eta = 0.1$ and $\eta = 0.5$.

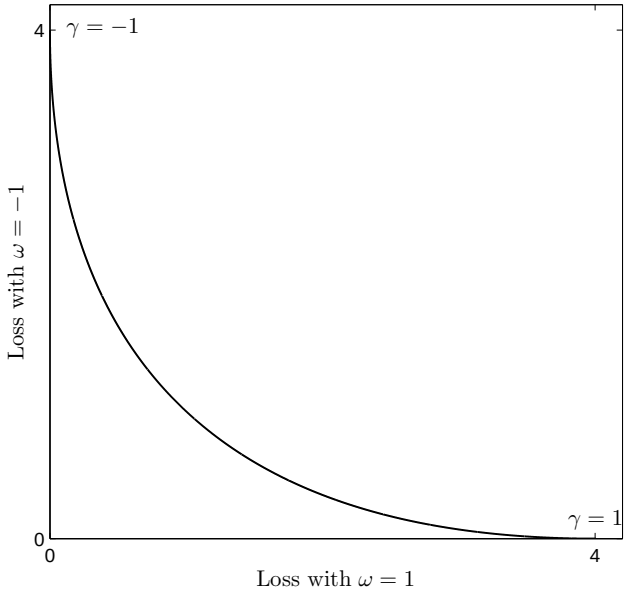


Figure 2.1: The parametric curve $((-1-\gamma)^2, (1-\gamma)^2)$ with $\gamma \in [-1, 1]$.

To find the values of η for which the curve is convex is equivalent to finding those values for which the second derivative of the curve is less or equal to zero for all values of $\gamma \in [-1, 1]$. We will first calculate the first derivative by the equation

$$\frac{dv}{du} = \frac{\frac{dv}{d\gamma}}{\frac{du}{d\gamma}} \quad .$$

The elements on the right hand side are given by

$$\frac{dv}{d\gamma} = 2\eta(1-\gamma)e^{-\eta(1-\gamma)^2} \quad , \text{ and}$$
$$\frac{du}{d\gamma} = -2\eta(1+\gamma)e^{-\eta(1+\gamma)^2} \quad .$$

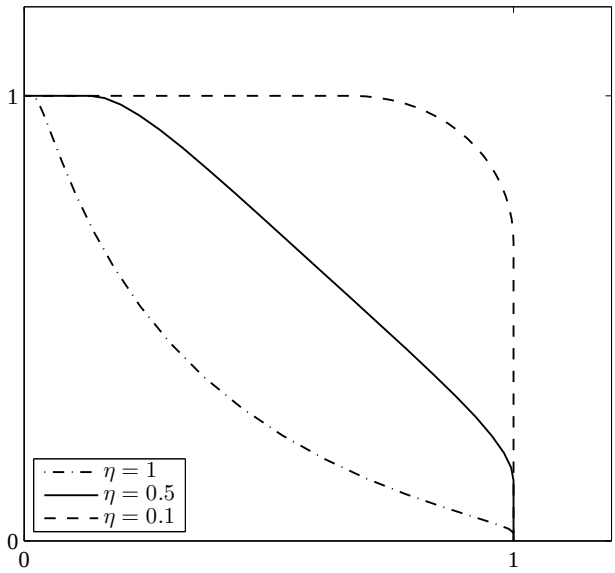


Figure 2.2: The parametric curve $\left(e^{-\eta(-1-\gamma)^2}, e^{-\eta(1-\gamma)^2}\right)$ for different values of η with $\gamma \in [-1, 1]$.

Therefore

$$\begin{aligned}\frac{dv}{du} &= \frac{2\eta(1-\gamma)e^{-\eta(1-\gamma)^2}}{-2\eta(1+\gamma)e^{-\eta(1+\gamma)^2}} \\ &= -\frac{1-\gamma}{1+\gamma}e^{-\eta(1-\gamma)^2+\eta(1+\gamma)^2} \\ &= -\frac{1-\gamma}{1+\gamma}e^{4\eta\gamma} \quad .\end{aligned}$$

To find the second derivative we will use

$$\frac{d^2v}{du^2} = \frac{\frac{d^2v}{d\gamma du}}{\frac{du}{d\gamma}} \quad .$$

We have already found $\frac{du}{d\gamma}$. For $\frac{d^2v}{d\gamma du}$ we get

$$\begin{aligned}\frac{d^2v}{d\gamma du} &= -\left(4\eta\frac{1-\gamma}{1+\gamma}e^{4\eta\gamma} - \frac{2}{(1+\gamma)^2}e^{4\eta\gamma}\right) \\ &= -\frac{e^{4\eta\gamma}}{(1+\gamma)^2}\left(4\eta(1-\gamma)(1+\gamma) - 2\right) \quad .\end{aligned}$$

Therefore

$$\frac{d^2v}{du^2} = \frac{e^{4\eta\gamma}}{(1+\gamma)^2}\left(\frac{4\eta(1-\gamma^2)-2}{2\eta(1+\gamma)e^{-\eta(1+\gamma)^2}}\right) \quad .$$

The only term that can make this negative is $4\eta(1-\gamma^2)-2$. Therefore $\frac{d^2v}{du^2} \leq 0$ if and only if

$$\begin{aligned}&4\eta(1-\gamma^2) - 2 \leq 0 \\ \implies\ &\eta \leq \frac{1}{2(1-\gamma^2)} \quad .\end{aligned}$$

Since $\gamma^2 \in [0,1]$, then $\frac{d^2v}{du^2} \leq 0$ for all γ iff

$$\eta \leq \frac{1}{2} \quad .$$

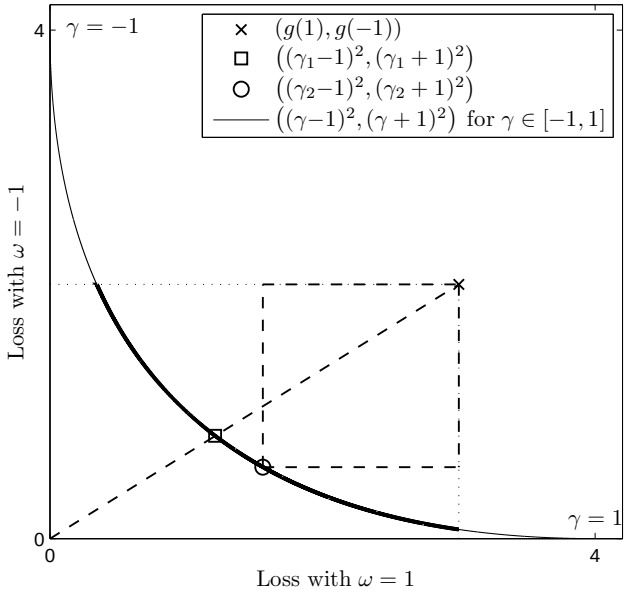


Figure 2.3: The parametric curve $((\gamma - 1)^2, (\gamma + 1)^2)$ for $\gamma \in [-1, 1]$.

Finding a Substitution Function

Recall that a substitution function Σ maps generalised predictions g to actual predictions γ. In Figure 2.3 we plot the parametric curve $((\gamma - 1)^2, (\gamma + 1)^2)$ for $\gamma \in [-1, 1]$ which shows the losses for all possible values of γ. The point $(g(1), g(-1))$ represents a generalised prediction. We want to find a corresponding point on the curve to be able to get an actual prediction. The bold part of the curve in Figure 2.3 represents the losses corresponding to all the predictions that a perfect substitution function is allowed to make given the generalised prediction.

An optimal substitution function for the restricted game would be one that gives us the γ corresponding to the point where the line $((0, 0), (g(1), g(-1))$ intersects with the losses curve. By optimal we mean that it attains the minimax of the ratio $\frac{g(\omega)}{\lambda(\omega,\gamma)}$. Let $x = g(1)$ and $y = g(-1)$. In Figure 2.3 this is the

point $((\gamma_1 - 1)^2, (\gamma_1 + 1)^2)$. Clearly, for this case we have

$$\frac{(\gamma_1 - 1)^2}{(\gamma_1 + 1)^2} = \frac{x}{y} \ .$$

This means that

$$\begin{aligned}
\sqrt{\frac{(\gamma_1 - 1)^2}{(\gamma_1 + 1)^2}} &= \sqrt{\frac{x}{y}} \\
-\frac{\gamma_1 - 1}{\gamma_1 + 1} &= \frac{\sqrt{x}}{\sqrt{y}} \\
\gamma_1 - 1 &= -\frac{\sqrt{x}\gamma_1}{\sqrt{y}} - \frac{\sqrt{x}}{\sqrt{y}} \\
\gamma_1 + \frac{\sqrt{x}\gamma_1}{\sqrt{y}} &= 1 - \frac{\sqrt{x}}{\sqrt{y}} \\
\gamma_1 \left(\frac{\sqrt{y} + \sqrt{x}}{\sqrt{y}} \right) &= \frac{\sqrt{y} - \sqrt{x}}{\sqrt{y}} \\
\gamma_1 &= \frac{\sqrt{y} - \sqrt{x}}{\sqrt{y} + \sqrt{x}} \ .
\end{aligned}$$

The negative sign on the left hand side of the second step appears because $(\gamma_1 - 1) \in [-2, 0]$ for all $\gamma_1 \in [-1, 1]$. The substitution function $\gamma_1 = \Sigma_1(g)$ is therefore

$$\gamma_1 = \frac{\sqrt{g(-1)} - \sqrt{g(1)}}{\sqrt{g(-1)} + \sqrt{g(1)}} \ . \tag{2.6}$$

Unfortunately, substitution function (2.6) is nonlinear and would prove difficult to use in practice. Therefore, we now attempt to find a simpler substitution function that finds the γ corresponding to the point on the losses curve that intersects with a square drawn from the generalised prediction. In Figure 2.3 this square is drawn with dashed lines and the corresponding point on the curve is $((\gamma_2 - 1)^2, (\gamma_2 + 1)^2)$. Once again, let $x = g(1)$ and $y = g(-1)$.

Since all sides of a square have equal length

$$\begin{aligned}(\gamma_2 - 1)^2 - x &= (\gamma_2 + 1)^2 - y \\ -2\gamma_2 - x &= 2\gamma_2 - y \\ \gamma_2 &= \frac{y - x}{4} \ .\end{aligned}$$

Therefore this substitution function is

$$\gamma_2 = \frac{g(-1) - g(1)}{4} \ . \tag{2.7}$$

2.3.2 Generalisation to the Full Game

The following lemma is an elaboration of the result in Haussler et al. [1998] that shows that the restriction $\omega_t \in \{-1, 1\}$ can be removed. It asserts that any substitution function for the restricted game is also a substitution function in the full game, where $\omega_t \in [-1, 1]$.

Lemma 3 (Vovk [2001, Lemma 3]) *Fix Y_1 and Y_2 such that $Y_1 < Y_2$. Let ω and p range over $[Y_1, Y_2]$ and $\mathbb{R}$ respectively, and $\lambda(\omega - p) = (\omega - p)^2$ be the square loss function. Let P be a probability distribution in $\mathbb{R}$, and let g be the generalised prediction corresponding to P, i.e.,*

$$g(\omega) = \log_\beta \int \beta^{(\omega - p)^2} P(dp) \ .$$

For every $\gamma \in \mathbb{R}$, if

$$\lambda(Y_1, \gamma) \le g(Y_1) \text{ and } \lambda(Y_2, \gamma) \le g(Y_2) \ ,$$

then

$$\lambda(\omega, \gamma) \le g(\omega) \ , \qquad \forall \omega \in [Y_1, Y_2] \ . \qquad \square$$

The Square Loss Game with $\omega_t \in [-Y, Y]$

In this section we generalise the results above for the square loss game with $\omega_t \in [-Y, Y]$ for any $Y \in \mathbb{R}$.

To find the values of η for which this game is η-mixable, let $\tilde{u} = e^{-\tilde{\eta}(-Y-\gamma)^2}$ where $\gamma \in [-Y, Y]$, corresponding to u in Section 2.3.1. We will reduce this to its equivalent in the full square loss game with $\omega_t \in [-1, 1]$ by using a scaling factor of Y^2. We get

$$\begin{aligned} u &= e^{-\tilde{\eta}\frac{(-Y-\gamma)^2}{Y^2}} \\ &= e^{-\frac{\tilde{\eta}}{Y^2}(-Y-\gamma)^2} \quad . \end{aligned}$$

Let $\eta = \frac{\tilde{\eta}}{Y^2}$. We know that for this game to be η-mixable $\tilde{\eta}$ has to be less or equal to $\frac{1}{2}$. Therefore, the full square loss game with $\omega_t \in [-Y, Y]$ is η-mixable if

$$\begin{aligned} \eta Y^2 &\leq \frac{1}{2} \\ \implies \eta &\leq \frac{1}{2Y^2} \quad . \end{aligned}$$

To get a substitution function for this game we simply repeat the procedure used to obtain (2.7) with $((\gamma - Y)^2, (\gamma + Y)^2)$ replacing $((\gamma - 1)^2, (\gamma + 1)^2)$, and $(g(Y), g(-Y))$ replacing $(g(1), g(-1))$. This gives us

$$\gamma = \frac{g(-Y) - g(Y)}{4Y} \quad . \tag{2.8}$$

Chapter 3

Online Regression

In this chapter we introduce the problem of online regression and some standard solutions. As usual, all vectors are identified with one-column matrices and $\mathbf{A}'$ stands for the transpose of matrix $\mathbf{A}$. We will not be specifying the size of simple matrices like the identity matrix $\mathbf{I}$ when this is clear from the context.

3.1 Protocol and Loss

In online regression at every moment in time $t = 1, 2, \ldots$, the value of a signal $\mathbf{x}_t \in X$ arrives. Statistician (or Learner) S observes $\mathbf{x}_t$ and then outputs a prediction $\gamma_t \in \mathbb{R}$. Finally, the outcome $y_t \in \mathbb{R}$ arrives. This can be summarised by the following scheme:

for $t = 1, 2, \ldots$ **do**
  S observes $\mathbf{x}_t \in X$
  S outputs $\gamma_t \in \mathbb{R}$
  S observes $y_t \in \mathbb{R}$
end for

The set X is a signal space which is assumed to be known to Statistician in advance. We will be referring to a signal-outcome pair as an example.

The performance of S is measured by the sum of square losses, which are the squared discrepancies between the predictions and the outcomes. Therefore,

on trial t Statistician S suffers loss $(y_t - \gamma_t)^2$. The losses incurred over several trials sum up to the overall loss, known as the cumulative square loss. Thus, after T trials the total loss of S is

$$\text{Loss}_T(S) = \sum_{t=1}^{T} (y_t - \gamma_t)^2 \ .$$

Clearly, a smaller value of $\text{Loss}_T(S)$ means a better predictive performance.

3.1.1 Batch Learning

Although our main interest is in the online mode of learning, the algorithms described in this dissertation can be naturally applied in batch learning mode. In this mode, Statistician S is given a training set comprised of ℓ signal-outcome pairs $(\mathbf{x}_i, y_i) \in X \times \mathbb{R}$ and a testing set containing new signals. S is required to output predictions that approximate the true outcomes of the signals in the testing set. Once again, the performance of S is measured by the square loss.

3.2 Linear and Kernel Predictors

If $X \subseteq \mathbb{R}^n$ we can consider simple linear predictors of the form $\theta \in \mathbb{R}^n$ that given a signal $\mathbf{x} \in X$ make a prediction $\theta'\mathbf{x}$. Linear methods are easy to manipulate mathematically but their use in the real world is limited since they can only model simple dependencies. One solution to this could be to map the data to some high dimensional feature space and then find a simple solution there. This, however, can lead to what is known as the curse of dimensionality where both the computational and generalisation performance degrade as the number of features grow [Cristianini and Shawe-Taylor, 2000, Section 3.1]. Kernels, defined below, can be used to make a linear algorithm operate in feature space without the inherent complexities.

Definition 2 (Kernels as Dot Products in Feature Space) Given a map-

ping $\phi : X \mapsto \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space, kernels are defined as

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \ . \qquad \square$$

Definition 3 (Reproducing Kernels of a Reproducing Kernel Hilbert Space (RKHS)) A Reproducing Kernel Hilbert Space (RKHS) on a set X is a Hilbert space $\mathcal{F}$ of real valued functions on X such that the evaluation functional $f \in \mathcal{F} \mapsto f(\mathbf{x})$ is continuous for each $\mathbf{x} \in X$. By the Riesz Representation theorem (see Lemma 7), for every $\mathbf{x} \in X$ there exists a function $k_{\mathbf{x}} \in \mathcal{F}$ such that

$$f(\mathbf{x}) = \langle k_{\mathbf{x}}, f \rangle \ ,$$

for all $f \in \mathcal{F}$. The reproducing kernel of $\mathcal{F}$ is the function $k : X \times X \mapsto \mathbb{R}$ such that

$$k(\mathbf{x}, \mathbf{z}) = \langle k_{\mathbf{x}}, k_{\mathbf{z}} \rangle = k_{\mathbf{x}}(\mathbf{z}) = k_{\mathbf{z}}(\mathbf{x}) \ . \qquad \square$$

Definition 4 (Kernels as Symmetric Positive Semi-Definite Functions) A kernel is any function $k : X \times X \mapsto \mathbb{R}$ that is symmetric

$$k(\mathbf{x}, \mathbf{z}) = k(\mathbf{z}, \mathbf{x})$$

for all $\mathbf{z}, \mathbf{x} \in X$, and positive semi-definite

$$\sum_{i,j=1}^{\ell} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for all $\ell \geq 1$, all $c_i, c_j \in \mathbb{R}$, and all $\mathbf{x}_i, \mathbf{x}_j \in X$. $\square$

These three definitions are equivalent since a function $k(\mathbf{x}, \mathbf{z}) : X \times X \mapsto \mathbb{R}$ can be represented in the form $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ iff k is the reproducing kernel of an RKHS iff k is symmetric and positive semi-definite.

Typically, to transform a linear method into a nonlinear one, the linear algorithm is first formulated in such a way that all signals appear only in dot products. This formulation is known as the dual form. These dot products are

then replaced by kernels. This procedure is known as the kernel trick and was first used in this context in Aizerman et al. [1964]. It follows from the definitions above that for all ℓ and all $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in X$ the kernel (or Gram) matrix $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$, $i, j = 1, \ldots, \ell$ is positive semi-definite, i.e., has nonnegative eigenvalues. In addition, for every kernel there exists a unique RKHS $\mathcal{F}$ such that k is the reproducing kernel of $\mathcal{F}$. Intuitively, if $D \in \mathcal{F}$, then $D(\mathbf{x})$ is a decision rule in $\mathcal{F}$ that produces a prediction for the object $\mathbf{x}$. We will be measuring the complexity of D by its norm $\|D\|$ in $\mathcal{F}$. For more information on kernels and RKHS see, for example, Aronszajn [1950], Schölkopf and Smola [2002, Chapter 2], and Vovk [2006, Sections 2 and 4].

3.2.1 Standard Kernels

There are several standard 'general purpose' kernels; choosing one (or creating a new one) depends on the task at hand. In the descriptions below, $\mathbf{x}, \mathbf{z} \in X$ are objects of dimension n. For more detailed information see, for example, Schölkopf and Smola [2002], Herbrich [2002] and Shawe-Taylor and Cristianini [2005].

The Linear Kernel

The linear kernel is the simplest of kernels since it is the standard dot product

$$k_l(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle \ .$$

Clearly, the mapping used is the identity function, therefore the input and feature spaces are the same.

The Polynomial Kernel

The polynomial kernel is a simple but powerful kernel given by

$$k_p(\mathbf{x}, \mathbf{z}) = (1 + \langle \mathbf{x}, \mathbf{z} \rangle)^d \ .$$

This kernel maps the elements of the vectors into the space spanned by all their monomials (products of features) up to and including the dth degree.

The Radial Basis Function Kernel

The radial basis function (RBF) kernel is calculated by

$$k_r(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right) \ .$$

RBF kernels map the input space onto the surface of an infinite dimensional unit hypersphere, because by construction $\|\phi(\mathbf{x})\| = \sqrt{k_r(\mathbf{x}, \mathbf{x})} = 1$ for all $\mathbf{x} \in X$. The parameter σ (the radius), which can be any positive real number, controls the amount of smoothing of the decision surface in input space. Big values of σ lead to a very flat and smooth decision surface and conversely, small values lead to a very convoluted decision surface that fits tightly around the given data.

The Spline Kernel

The spline kernel is what is known as a multiplicative kernel, where the multi-dimensional case is achieved by taking the product of the one-dimensional case. The one-dimensional linear spline kernel with an infinite number of nodes is

$$k_{s1}(x, z) = \frac{\min(x, z)^3}{3} + \frac{\min(x, z)^2|x - z|}{2} + xz + 1 \ . \tag{3.1}$$

Clearly, the n-dimensional case is calculated by

$$k_s(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} k_{s1}(x_i, z_i) \ ,$$

where x_i denotes the ith element of object $\mathbf{x}$. The spline kernel requires that all the elements of its arguments are positive.

The ANOVA Kernel

ANOVA (ANalysis Of VAriance) is a statistical technique used to analyse the interactions between attributes. Given a one-dimensional multiplicative kernel k_1, the ANOVA kernel of order d is defined as

$$k_a^{(d)}(\mathbf{x}, \mathbf{z}) = \sum_{1 \leq i_1 < \ldots < i_d \leq n} k_1(x_{i_1}, z_{i_1}) \times \cdots \times k_1(x_{i_d}, z_{i_d}) \ ,$$

where x_{i_j} and z_{i_j} are the i_jth elements of vectors $\mathbf{x}$ and $\mathbf{z}$ respectively.

We now give a recurrent procedure to calculate this kernel as described in Burges and Vapnik [1995] and Stitson et al. [1999]. Let $k_a^{(0)}(\mathbf{x}, \mathbf{z}) = 1$, and once again, let k_1 be a one-dimensional multiplicative kernel. The ANOVA kernel of order d can be computed by

$$k_a^{(d)}(\mathbf{x}, \mathbf{z}) = \frac{1}{d} \sum_{i=1}^{d} (-1)^{i+1} k_a^{(d-i)}(\mathbf{x}, \mathbf{z}) \sum_{j=1}^{n} \left(k_1(x_j, z_j)\right)^i \ ,$$

where x_j and z_j are the jth elements of vectors $\mathbf{x}$ and $\mathbf{z}$ respectively. This recurrent procedure can be implemented very efficiently by using a dynamic programming technique where all lower orders are calculated at the same time.

There are two ways of using ANOVA decomposition to produce kernels of order d. The first method includes order d and all lower orders,

$$k_a(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{d} k_a^{(d)}(\mathbf{x}, \mathbf{z}) \ .$$

The second method only includes order d,

$$k_a(\mathbf{x}, \mathbf{z}) = k_a^{(d)}(\mathbf{x}, \mathbf{z}) \ .$$

Following Stitson et al. [1999], in our experiments we use ANOVA decompositions of the latter type, that is, considering the term of order d only. When this kernel is applied to (3.1), i.e., $k_1 = k_{s1}$, we get the ANOVA spline kernel.

3.3 Existing Solutions

Let us model the data by the linear equation

$$y_t = \langle \mathbf{w}, \mathbf{x}_t \rangle + \varepsilon_t \ , \tag{3.2}$$

where $\mathbf{w}, \mathbf{x}_t \in \mathbb{R}^n$ and $\varepsilon_t \in \mathbb{R}$ is some noise. The most popular solutions to this problem are Least Squares (LS) and Ridge Regression (RR). LS finds the solution that best fits the data, while RR balances the goodness of fit of the solution with its complexity.

3.3.1 Least Squares

The method of Least Squares (LS) was derived independently by Legendre and Gauss in 1805 and 1809 respectively. At time T it aims to find a solution $\mathbf{w}_{\text{L}}$ to (3.2) that minimises the overall sum of square losses over the previously seen examples[1]

$$\mathcal{L}_T(\text{LS}) = \sum_{t=1}^{T-1} (y_t - \langle \mathbf{w}_{\text{L}}, \mathbf{x}_t \rangle)^2 \ . \tag{3.3}$$

If we let $\mathbf{w} = \mathbf{w}_{\text{L}}$ and formulate (3.3) in matrix notation we get

$$\begin{aligned} \mathcal{L}_T(\text{LS}) &= (\mathbf{y} - \mathbf{X}\mathbf{w})^2 \\ &= \mathbf{y}'\mathbf{y} - 2\mathbf{w}'\mathbf{X}'\mathbf{y} + \mathbf{w}'\mathbf{X}'\mathbf{X}\mathbf{w} \ , \end{aligned}$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{T-1})'$ and $\mathbf{y} = (y_1, \ldots, y_{T-1})'$. To find the $\mathbf{w}$ that minimises this we take its first derivative

$$\frac{\partial \mathcal{L}_T(\text{LS})}{\partial \mathbf{w}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} \ . \tag{3.4}$$

[1] It may seems strange that in (3.3) (and other places) the size of the training set is $T-1$. This is indeed unusual but is given in this form to be consistent with the other methods' objective functions.

The minimum is attained when (3.4) equals 0, therefore,

$$\begin{aligned} 0 &= -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} \\ 2\mathbf{X}'\mathbf{X}\mathbf{w} &= 2\mathbf{X}'\mathbf{y} \\ \implies \mathbf{w}_{\text{L}} &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \ . \end{aligned}$$

3.3.2 Ridge Regression

Least Squares runs into problems when some features in $\mathbf{X}$ are highly correlated because the matrix $\mathbf{X}'\mathbf{X}$ becomes close to singular, resulting in unstable solutions. Ridge Regression (RR), first introduced to statistics in Hoerl [1962], differs from Least Squares in that at time T its objective is to minimise

$$\mathcal{L}_T(\text{RR}) = a\|\mathbf{w}_{\text{R}}\|^2 + \sum_{t=1}^{T-1}(y_t - \langle \mathbf{w}_{\text{R}}, \mathbf{x}_t \rangle)^2 \ , \tag{3.5}$$

where a is a fixed nonnegative real number. To find the solution to this we will take its derivative and set it equal to 0 similar to what we did for Least Squares. Letting $\mathbf{w} = \mathbf{w}_{\text{R}}$ and using matrix notation, (3.5) becomes

$$\begin{aligned} \mathcal{L}_T(\text{RR}) &= a(\mathbf{w}'\mathbf{w}) + (\mathbf{y} - \mathbf{X}\mathbf{w})^2 \\ &= a(\mathbf{w}'\mathbf{w}) + \mathbf{y}'\mathbf{y} - 2\mathbf{w}'\mathbf{X}'\mathbf{y} + \mathbf{w}'\mathbf{X}'\mathbf{X}\mathbf{w} \ . \end{aligned}$$

If we differentiate this with respect to $\mathbf{w}$, divide throughout by 2 and set it equal to 0 we get

$$\frac{1}{2}\frac{\partial \mathcal{L}_T(\text{RR})}{\partial \mathbf{w}} = a\mathbf{w} - \mathbf{X}'\mathbf{y} + \mathbf{X}'\mathbf{X}\mathbf{w} = 0 \ .$$

Making $\mathbf{w} = \mathbf{w}_{\text{R}}$ subject of the formula gives us

$$\mathbf{w}_{\text{R}} = (a\mathbf{I} + \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \ , \tag{3.6}$$

where $\mathbf{I}$ is the identity matrix.

Dual (Kernel) Form

By using Lemma 4 we can get a dual form of RR's prediction formula which makes a prediction for the new signal $\mathbf{x}_T$ by

$$\gamma_{\text{RR}} = \mathbf{y}'(a\mathbf{I} + \mathbf{X}\mathbf{X}')^{-1}\mathbf{X}\mathbf{x}_T \ . \tag{3.7}$$

The kernel version of this, referred to as Kernel Ridge Regression (KRR) (see Saunders et al. [1998]), obtained by replacing all dot products with kernels is

$$\gamma_{\text{KRR}} = \mathbf{y}'(a\mathbf{I} + \mathbf{K})^{-1}\mathbf{k} \ , \tag{3.8}$$

where $\mathbf{k} = (k(\mathbf{x}_i, \mathbf{x}_T))$, for $i = 1, \ldots, T-1$, and $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$, for $i, j = 1, \ldots, T-1$.

3.3.3 Comparison of Least Squares and Ridge Regression

Clearly, Ridge Regression (RR) is a generalisation of Least Square (LS), because LS is RR with $a = 0$. On the other hand, RR is equivalent to LS with n fictitious examples added to its training data, where n is the dimensionality of the data. A signal $\mathbf{a}_i$ of these fictitious examples is all zero except for the ith element which is equal to $\sqrt{a}$. If we transpose all these signals and put them in a matrix under each other (similar to what we do to get $\mathbf{X}$) we get the $n \times n$ matrix

$$\mathbf{A} = \sqrt{a}\mathbf{I} = \begin{bmatrix} \sqrt{a} & 0 & \cdots & 0 \\ 0 & \sqrt{a} & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{a} \end{bmatrix} \ .$$

Let the outcomes of all these fictitious examples be equal to 0. To see how RR is LS with n fictitious training examples, consider the following:

$$\begin{aligned}\begin{bmatrix} \mathbf{X} \\ \mathbf{A} \end{bmatrix}' \begin{bmatrix} \mathbf{X} \\ \mathbf{A} \end{bmatrix} &= \mathbf{X}'\mathbf{X} + \mathbf{A}'\mathbf{A} \\ &= \mathbf{X}'\mathbf{X} + a\mathbf{I} \ .\end{aligned}$$

Clearly, each one of these fictitious examples pushes the corresponding element of the solution vector $\mathbf{w}_{\text{R}}$ towards 0 by an amount proportional to a. Effectively, this decreases the norm of $\mathbf{w}_{\text{R}}$, reaching the aim of Ridge Regression's objective function (3.5) by favouring a $\mathbf{w}_{\text{R}}$ with smaller elements. This regularisation reduces the complexity of the solution, decreasing the risk of overfitting the training data, and consequently leads to better generalisation. A related effect is that having $a > 0$ stabilises the solution since this makes the matrix $(a\mathbf{I} + \mathbf{X}'\mathbf{X})$ positive definite and therefore nonsingular.

We now show a toy example where this regularisation benefits Ridge Regression. Consider the cubic polynomial

$$y = -0.5x^3 + x^2 + 100x \ , \tag{3.9}$$

where $x \in \mathbb{R}$. We want to predict (3.9) using LS and RR from 4 training points that have been corrupted by noise. In Figure 3.1 we show the results obtained. Clearly, RR with $a = 0.1$ approximates the curve better than LS which is equivalent to RR with $a = 0$. From the figure it is clear that LS overfits the training data, resulting in a bad generalisation performance. For completeness, we also included RR's predictions when $a = 1$ in the figure to show the effect of over regularisation. If we take points from the interval $[-10, 10]$ at steps of 0.1, we get 201 outcomes and predictions per method. Using these values, the mean square loss of LS is 5.62×10^3, while that of RR with $a = 0.1$ is 3.27×10^3. The mean square loss of the over regularised RR is 3.56×10^4.

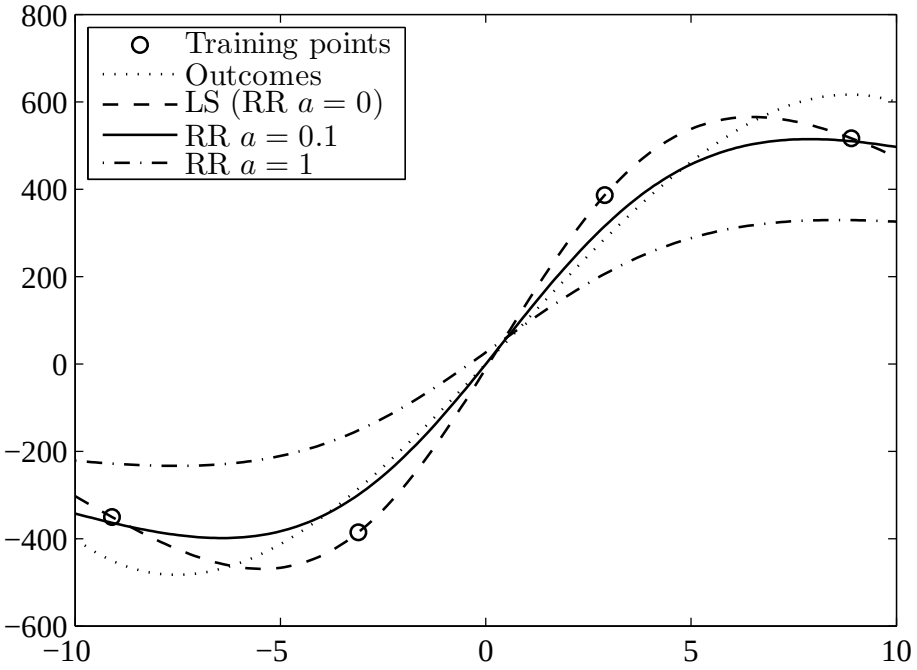


Figure 3.1: Least Squares (LS) and Ridge Regression (RR) approximating a cubic polynomial from noisy data.

Chapter 4

Improving the Aggregating Algorithm for Regression

As its name suggests, the Aggregating Algorithm for Regression (AAR) is an application of the Aggregating Algorithm (AA) to the problem of regression. KAAR is the kernel version of AAR. AAR and KAAR have very interesting theoretical properties; however, in this chapter we show that KAAR's empirical performance is, in general, worse than that of the popular Kernel Ridge Regression (KRR). We find that KAAR is better only when the data is corrupted by a lot of noise or contains severe outliers. In an effort to create methods that perform well whether the data is corrupted or not, we introduce mainly two new hybrid methods. Empirical results suggest that our new methods perform better than KAAR on corrupted datasets and comparably to KRR on regular data. In this chapter we also give a new Bayesian interpretation of KAAR and our hybrid methods.

4.1 Introduction

As we saw in Chapter 3, in regression we are interested in learning a relationship between a signal, which can consist of one or more independent variables, and its outcome. In the simplest of models this relationship is taken to be linear, but nonlinear relationships are common in nature. Once this relationship is established it is possible to predict the outcomes of previously unseen

signals.

Ridge Regression (RR) (see Section 3.3.2) attempts to balance the solution's goodness of fit on the data with the size of its complexity. Any method that prevents overfitting of the data is known as regularisation. RR works very well on real world data and is still very popular today. The Aggregating Algorithm for Regression (AAR) [Vovk, 2001] (see also the Vovk-Azoury-Warmuth algorithm in Cesa-Bianchi and Lugosi [2006, Section 11.8]) is a relatively new method and is shown to be only a little worse than any linear predictor in the online mode of learning. It happens that AAR is similar to RR but with some extra regularisation added.

Similarly to Ridge Regression, AAR was formulated in dual variables to get the Kernel Aggregating Algorithm for Regression (KAAR). A general worst case upper bound on its loss (in the online context) that does not hold for KRR was derived [Gammerman et al., 2004, Vovk, 2001]. KRR is optimal only under some probabilistic assumptions (see Section 4.4.1 and Vovk et al. [2005]), while KAAR's bound does not make any assumptions on the underlying probability distribution of the data. This makes KAAR theoretically applicable to a much wider group of datasets. In particular, this bound does not require the data to be independently identically distributed (i.i.d.). In many practical applications i.i.d. is unrealistic to assume. AAR and its kernel version are described in Section 4.2.

In empirical experiments KAAR performs better than KRR when the data is corrupted by lots of noise or contains severe outliers. However, this is not true for regular datasets with KAAR suffering more loss. This happens because of KAAR's extra regularisation compared to KRR. Therefore, in Section 4.3 we introduce new hybrid methods, primarily Iterative KAAR (IKAAR) and Controlled KAAR (CKAAR). These methods modify KAAR in such a way as to be able to control the amount of extra regularisation, the choice of which should depend on the data at hand. A comparison of all the methods introduced and a Bayesian interpretation is given in Section 4.4. Surprisingly, KAAR and our methods can be seen as pushing KRR's prediction towards the mean of the outcomes by an amount proportional to the variance of the prediction itself.

Finally, in Section 4.5 we report the empirical performance of all these methods on the Gaze dataset [Quiñonero-Candela et al., 2006] and the Boston Housing dataset [Asuncion and Newman, 2007]. The Gaze dataset is of particular interest to us since it is known to contain severe outliers. On this dataset our methods perform significantly better than Kernel Ridge Regression. On more regular datasets, like the Boston Housing dataset, KAAR suffers more loss than KRR while our hybrid methods perform comparably to KRR.

4.2 The Aggregating Algorithm for Regression (AAR)

As we saw in Chapter 2, the Aggregating Algorithm (AA) [Vovk, 1990] is a technique that makes predictions using expert advice. This means that AA observes the next signal in a sequence and also the predictions of a (possibly infinite) pool of experts. It then merges the experts' predictions and outputs its own prediction which is in a sense optimal. AA was applied to the problem of linear regression resulting in the Aggregating Algorithm for Regression (AAR) which merges all the linear predictors that map signals to outcomes [Vovk, 2001].

The AAR solution to the regression problem is

$$\mathbf{w}_{\mathrm{A}} = (a\mathbf{I} + \widetilde{\mathbf{X}}'\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}'\widetilde{\mathbf{y}} \ , \tag{4.1}$$

where $a > 0$, $\widetilde{\mathbf{X}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)'$ and $\widetilde{\mathbf{y}} = (y_1, y_2, \ldots, y_{T-1}, 0)'$. We will now show that at time T AAR finds a solution $\mathbf{w}_{\mathrm{A}}$ that minimises

$$\mathcal{L}_T(\mathrm{AAR}) = a\|\mathbf{w}_{\mathrm{A}}\|^2 + \langle \mathbf{w}_{\mathrm{A}}, \mathbf{x}_T \rangle^2 + \sum_{t=1}^{T-1} (y_t - \langle \mathbf{w}_{\mathrm{A}}, \mathbf{x}_t \rangle)^2 \ . \tag{4.2}$$

If for simplicity of notation we let $\mathbf{w} = \mathbf{w}_{\mathrm{A}}$ and formulate (4.2) in matrix

notation we get

$$\begin{aligned}\mathcal{L}_T(\mathrm{AAR}) &= a(\mathbf{w}'\mathbf{w}) + (\mathbf{w}'\mathbf{x}_T)^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})^2 \\ &= a(\mathbf{w}'\mathbf{w}) + \widetilde{\mathbf{y}}'\widetilde{\mathbf{y}} - 2\mathbf{w}'\widetilde{\mathbf{X}}'\widetilde{\mathbf{y}} + \mathbf{w}'\widetilde{\mathbf{X}}'\widetilde{\mathbf{X}}\mathbf{w} \ .\end{aligned}$$

If we differentiate this with respect to $\mathbf{w}$, divide throughout by 2 and set it equal to 0 we get

$$\frac{1}{2}\frac{\partial \mathcal{L}_T(\mathrm{AAR})}{\partial \mathbf{w}} = a\mathbf{w} - \widetilde{\mathbf{X}}'\widetilde{\mathbf{y}} + \widetilde{\mathbf{X}}'\widetilde{\mathbf{X}}\mathbf{w} = 0 \ .$$

Making $\mathbf{w} = \mathbf{w}_{\mathrm{A}}$ subject of the formula gives us (4.1).

AAR assumes that outcomes are bounded, that is, that they come from the interval $[-Y, Y]$, $Y \in \mathbb{R}$; however, note that the algorithm does not need to know the value of Y. The main property of AAR is that the total loss it suffers is only a little worse than that of any linear predictor. By the latter we mean a strategy that predicts $\theta'\mathbf{x}_t$ on every trial t, where $\theta \in \mathbb{R}^n$ is some fixed vector. The set of all linear predictors may be identified with $\mathbb{R}^n$.

Theorem 2 (Vovk [2001, Theorem 1]) *For every positive integer n, any $a > 0$, and every point in time T,*

$$\mathrm{Loss}_T(AAR) \le \inf_\theta(\mathrm{Loss}_T(\theta) + a\|\theta\|^2) + Y^2 \ln\det\left(\mathbf{I} + \frac{1}{a}\sum_{t=1}^{T}\mathbf{x}_t\mathbf{x}_t'\right) \ . \quad (4.3)$$

□

It is interesting to note that AAR's bound does not make any assumptions on the probability distribution of the data. From (4.1) it is clear that in computational terms AAR is similar to Ridge Regression but with the signal-outcome pair $(\mathbf{x}_T, 0)$ added to its training set, where $\mathbf{x}_T$ is the new signal for which a prediction is to be made. This makes predictions shrink towards 0 with the goal of making them even more resistant to overfitting (it is assumed that the mean of the outcomes is 0).

4.2.1 The Kernel Aggregating Algorithm for Regression (KAAR)

AAR has interesting theoretical properties; however, its use in the real world is limited since it can only model simple linear dependencies. In Gammerman et al. [2004] AAR was formulated in dual variables to be able to introduce nonlinearity through kernels. It follows directly from (4.1) and Lemma 4 that AAR's prediction in dual variables is

$$\begin{aligned}\gamma_{\text{AAR}} &= \mathbf{w}_{\text{A}}'\mathbf{x} \\ &= \left(\left(a\mathbf{I} + \widetilde{\mathbf{X}}'\widetilde{\mathbf{X}}\right)^{-1}\widetilde{\mathbf{X}}'\widetilde{\mathbf{y}}\right)'\mathbf{x} \\ &= \widetilde{\mathbf{y}}'\left(a\mathbf{I} + \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}'\right)^{-1}\widetilde{\mathbf{X}}\mathbf{x} \ .\end{aligned}$$

Notice that now all signals appear only in dot products. To get the kernel version of AAR, which we shall call the Kernel Aggregating Algorithm for Regression (KAAR), we simply replace these dot products with kernels. KAAR's prediction for the signal $\mathbf{x}_T$ is therefore calculated by

$$\gamma_{\text{KAAR}} = \widetilde{\mathbf{y}}'(a\mathbf{I} + \widetilde{\mathbf{K}})^{-1}\widetilde{\mathbf{k}} \ , \tag{4.4}$$

where

$$\widetilde{\mathbf{K}} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_{T-1}) & k(\mathbf{x}_1,\mathbf{x}_T) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdots & k(\mathbf{x}_2,\mathbf{x}_{T-1}) & k(\mathbf{x}_2,\mathbf{x}_T) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k(\mathbf{x}_{T-1},\mathbf{x}_1) & k(\mathbf{x}_{T-1},\mathbf{x}_2) & \cdots & k(\mathbf{x}_{T-1},\mathbf{x}_{T-1}) & k(\mathbf{x}_{T-1},\mathbf{x}_T) \\ k(\mathbf{x}_T,\mathbf{x}_1) & k(\mathbf{x}_T,\mathbf{x}_2) & \cdots & k(\mathbf{x}_T,\mathbf{x}_{T-1}) & k(\mathbf{x}_T,\mathbf{x}_T) \end{bmatrix} ,$$

and

$$\widetilde{\mathbf{k}} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_T) \\ k(\mathbf{x}_2, \mathbf{x}_T) \\ \vdots \\ k(\mathbf{x}_{T-1}, \mathbf{x}_T) \\ k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} \quad .$$

KAAR performs little worse than any decision rule D in the RKHS induced by a kernel function k. The following theorem generalises Theorem 2. Recall that AAR and therefore KAAR, assume that outcomes come from the interval $[-Y, Y]$.

Theorem 3 (Gammerman et al. [2004, Theorem 1]) *Let k be a kernel on a space X and D be any decision rule in the RKHS $\mathcal{F}$ induced by k. Then for every $a > 0$ and any point in time T the following holds:*

$$\mathrm{Loss}_T(KAAR) \leq \mathrm{Loss}_T(D) + a\|D\|^2 + Y^2 \ln \det \left(\mathbf{I} + \frac{1}{a} \widetilde{\mathbf{K}} \right) \quad . \tag{4.5}$$

□

Estimating the determinant of a positive definite matrix by the product of its diagonal elements (see Beckenbach and Bellman [1961, Section 2.10, Theorem 7]) and using the inequality $\ln(1 + x) \leq x$ we get the following Corollary.

Corollary 1 (Vovk [2006, Section 8]) *Under the conditions of Theorem 3 let $u = \sup_{\mathbf{x} \in X} \sqrt{k(\mathbf{x}, \mathbf{x})}$. Then for every $a > 0$, every $d > 0$, every decision rule D such that $\|D\| \leq d$ and any point in time T, we get*

$$\mathrm{Loss}_T(KAAR) \leq \mathrm{Loss}_T(D) + ad^2 + \frac{Y^2 u^2 T}{a} \quad .$$

If, moreover, T is known in advance, it is possible to minimise this by taking $a = (Yu/d)\sqrt{T}$ to get

$$\begin{aligned} \mathrm{Loss}_T(KAAR) &\leq \mathrm{Loss}_T(D) + 2Yud\sqrt{T} \\ &= \mathrm{Loss}_T(D) + o(T) \quad . \end{aligned}$$

□

4.3 Improving the Empirical Performance of KAAR

It is shown in Vovk [2001, Theorem 3] that AAR's bound does not hold for Ridge Regression (RR) therefore AAR has a better theoretical worst case performance bound than RR. On the other hand RR can be shown to be a Bayesian method (see Section 4.4.1). This means that under certain probabilistic assumptions on the data (namely independence and normally distributed noise), RR has optimal properties on average. Although we cannot realistically expect these assumptions to hold for real world datasets, RR is known to perform very well in practice. In fact, on most datasets KRR suffers less loss than KAAR. However, there are instances where KAAR performs better, for example when the data is heavily corrupted with noise or has severe outliers, such as in the Gaze dataset [Quiñonero-Candela et al., 2006] as shown in Section 4.5.

Figure 4.1 shows the predictions of KRR and KAAR on a test set containing 25 signals from a particular permutation[1] of the Boston Housing dataset [Asuncion and Newman, 2007]. Note that the signals in the test set have been sorted by their target outcome and that the x-axis represents the number of a signal and is not the signal itself (which is a vector with 13 features). This sorting was done exclusively to make the figure clearer. Moreover, note that KRR's prediction is negative in two instances. This does not make any sense as far as house prices are concerned; however, for fairness we do not truncate such results for any method. In this example the mean square loss of KRR is approximately 26.64 while that of KAAR is approximately 29.33. This means that KRR's performance is better. However, analysis of the individual predictions reveals that 44% of KAAR's predictions are more accurate. Through this and other empirical experiments it became evident that many times KAAR's predictions are overly rigid while KRR's predictions sometimes fluctuate too much. It was also observed that occasionally a better prediction can be somewhere in between those of KRR and KAAR.

[1] For a different permutation of the dataset the figure will be different but the general idea of what we are trying to show holds.

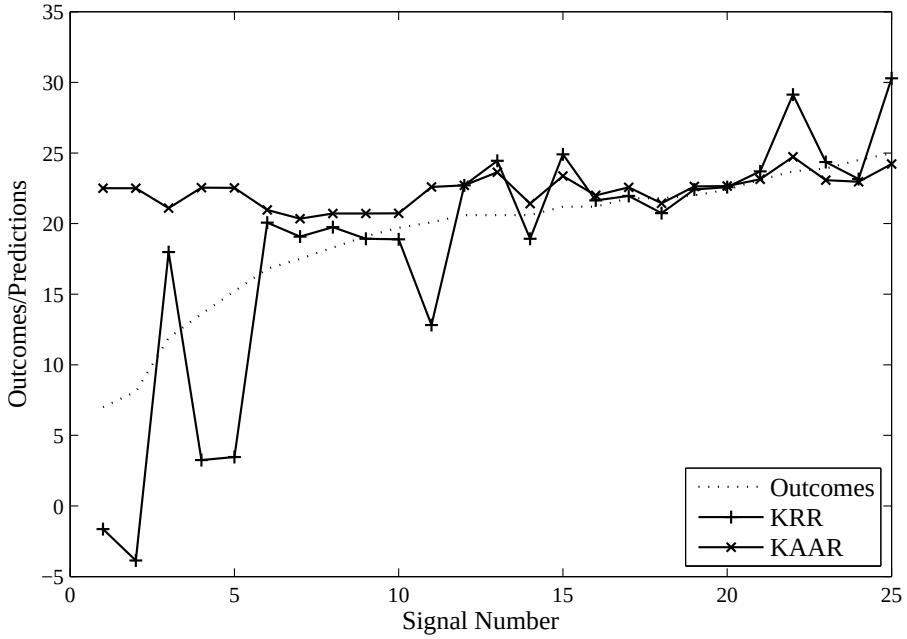


Figure 4.1: KRR and KAAR approximating a signal-outcome behaviour.

Equation (3.8) on page 40 and (4.4) show that KRR and KAAR are rather similar from a computational perspective. Is it possible therefore to combine these two methods to give a new method that in general is more accurate than both? Below we present three new methods that attempt to achieve this. The first method that we propose (KOKO) is a simple convex combination of KRR's and KAAR's predictions. It is included here for completeness and for comparison reasons. The second method, which we call Iterative KAAR (IKAAR), modifies the KAAR algorithm so that it outputs a sequence of predictions for a particular signal. We show that this sequence starts from the KAAR prediction and converges towards the prediction of KRR giving us a smooth transition from the former to the latter. The third method that we propose uses the fact that in Figure 4.1 KRR seems to fluctuate a lot with a standard deviation of 8.87, while KAAR is overly rigid having a standard deviation of 1.14 (the standard deviation of the real outcomes is 4.78). We therefore modify KAAR's objective to give us a new method where we can control the rigidness of the predictions. The resulting method, Controlled KAAR (CKAAR), has a parameter that allows it to change its behaviour. For two particular values of this parameter CKAAR is equivalent to KRR and KAAR.

Unfortunately, it is unclear whether we can have theoretical upper bounds on the square losses of these new methods. This is because they are really hybrids between KRR and KAAR which are motivated by very different theoretical backgrounds.

4.3.1 Simple Convex Combination (KOKO)

One of the simplest ways of combining KRR and KAAR is to take a convex combination of their predictions. This new 'method', which we have dubbed KOKO, makes its predictions as follows:

$$\gamma_{\text{KOKO}} = (1-\theta)\gamma_{\text{KRR}} + \theta\gamma_{\text{KAAR}} \ , \tag{4.6}$$

where θ is a scalar from the interval $[0, 1]$.

4.3.2 Iterative KAAR (IKAAR)

As we saw in Section 4.2, KAAR is equivalent to KRR with the signal-outcome pair $(\mathbf{x}_T, 0)$ added to its training set, where $\mathbf{x}_T$ is the new signal. Having 0 as the signal's outcome added to the training set pushes the prediction towards 0 and is what makes KAAR's predictions so rigid. In order to alleviate this we propose a new method, the Iterative Kernel Aggregating Algorithm for Regression (IKAAR). In its first iteration IKAAR is equivalent to KAAR in that it adds the pair $(\mathbf{x}_T, 0)$ to its training set. This produces the prediction γ_{KAAR}. However, in its second iteration IKAAR replaces the extra pair in its training set with a new pair $(\mathbf{x}_T, \gamma_{\text{KAAR}})$. This produces another prediction that in turn is used to replace γ_{KAAR} in the training set to make a new prediction. This procedure can be repeated an arbitrary number of times resulting in a sequence of IKAAR predictions for the same signal. We will denote these predictions by $\gamma^{(m)}_{\text{IKAAR}}$ where the index (m) denotes the iteration number. For clarity of notation let $\gamma^{(m)} = \gamma^{(m)}_{\text{IKAAR}}$ and $\mathbf{x} = \mathbf{x}_T$. We define IKAAR more formally as follows:

$$\gamma^{(m)} = \widetilde{\mathbf{y}}^{(m)\prime}(a\mathbf{I} + \widetilde{\mathbf{K}})^{-1}\widetilde{\mathbf{k}} \ , \tag{4.7}$$

where $\gamma^{(0)} = 0$, $m \geq 1$, and $\widetilde{\mathbf{y}}^{(m)} = \left(\mathbf{y}', \gamma^{(m-1)}\right)'$. Note that towards the end of this section and in Section 4.4 we give an explicit formula that computes $\gamma^{(m)}$ directly for any m.

Theorem 4 *For any signal, IKAAR's predictions start from the KAAR prediction and converge towards that of KRR as the number of IKAAR iterations approaches infinity.* □

Proof It follows from IKAAR's definition that the first prediction $\gamma^{(1)}$ is equivalent to KAAR's prediction. We will now show that IKAAR's predictions for any signal converge towards that of KRR as m approaches infinity. We can open up (4.7) in the following way:

$$\gamma^{(m)} = \begin{bmatrix} \mathbf{y} \\ \gamma^{(m-1)} \end{bmatrix}' \begin{bmatrix} \mathbf{K} + a\mathbf{I} & \mathbf{k} \\ \mathbf{k}' & k(\mathbf{x},\mathbf{x}) + a \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{k} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix} \ , \tag{4.8}$$

where $\mathbf{y}$, $\mathbf{K}$, and $\mathbf{k}$ are as in Section 3.3, and k is the kernel function. This

equation shows explicitly how $\gamma^{(m-1)}$ is being modified to get $\gamma^{(m)}$. We shall show that this transformation of $\gamma^{(m-1)}$ can be characterised by the linear equation

$$\gamma^{(m)} = s\gamma^{(m-1)} + c \ , \tag{4.9}$$

where $s, c \in \mathbb{R}$. If we manage to show that $0 \leq |s| < 1$ then it would follow from the Banach fixed point theorem (see Lemma 5) that IKAAR's predictions converge to a fixed point r, such that $r = sr + c$. Therefore, as $m \to \infty$, then $\gamma^{(m-1)} \to \gamma^{(m)}$ and $\gamma^{(m)} \to r$.

If we do the inversion in (4.8) by partitioning (see Lemma 6), we get

$$\begin{aligned}
\gamma^{(m)} &= \begin{bmatrix} \mathbf{y} \\ \gamma^{(m-1)} \end{bmatrix}' \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{Q} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{y} \\ \gamma^{(m-1)} \end{bmatrix}' \begin{bmatrix} \widetilde{\mathbf{P}} & \widetilde{\mathbf{Q}} \\ \widetilde{\mathbf{R}} & \widetilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{Q} \\ k(\mathbf{x},\mathbf{x}) \end{bmatrix} \\
&= \left(\widetilde{\mathbf{R}}\mathbf{Q} + \widetilde{\mathbf{S}}k(\mathbf{x},\mathbf{x})\right)\gamma^{(m-1)} + \left(\mathbf{y}'\widetilde{\mathbf{P}}\mathbf{Q} + \mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x},\mathbf{x})\right) \ ,
\end{aligned} \tag{4.10}$$

where $\mathbf{P} = \mathbf{K} + a\mathbf{I}$, $\mathbf{Q} = \mathbf{R}' = \mathbf{k}$, and $\mathbf{S} = k(\mathbf{x},\mathbf{x}) + a$ (in this case all the necessary inverses exist). Note that (4.10) is in the form (4.9). Therefore, if we make substitutions for $\widetilde{\mathbf{P}}$, $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{R}}$, and $\widetilde{\mathbf{S}}$ in (4.10) we get

$$s = \frac{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}}{k(\mathbf{x},\mathbf{x}) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k} + a} \ , \text{ and} \tag{4.11}$$

$$c = \mathbf{y}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}(1 - s) \ . \tag{4.12}$$

We will now proceed to show that s is always in the interval $[0, 1)$. Since by definition $a > 0$, we only need to show that $k(\mathbf{x},\mathbf{x}) \geq \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}$ to reach our goal. We will first show this for the linear kernel (the dot product) and subsequently we will generalise the result for the nonlinear kernel case. Therefore, for the linear kernel we have to show that for every $\mathbf{x}$ the following holds (the second line follows from Lemma 4):

$$\mathbf{x}'\mathbf{x} \geq (\mathbf{X}\mathbf{x})'(\mathbf{X}\mathbf{X}' + a\mathbf{I})^{-1}\mathbf{X}\mathbf{x} \tag{4.13}$$

$$= \mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}\mathbf{x} \ . \tag{4.14}$$

In order to do this we will first reduce (4.14) to a simpler form. Since $\mathbf{X}'\mathbf{X}$ is symmetric it can be diagonalised so that $\mathbf{X}'\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$, where the columns of the unitary matrix $\mathbf{V}$ are the eigenvectors of $\mathbf{X}'\mathbf{X}$ and $\mathbf{\Lambda}$ is the diagonal matrix made up of the corresponding eigenvalues λ_i. Recall that since $\mathbf{V}$ is a unitary matrix, $\mathbf{V}^{-1} = \mathbf{V}'$, so $\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I}$.

Performing the substitution $\mathbf{x} = \mathbf{V}\mathbf{z}$ (where $\mathbf{z} \in \mathbb{R}^n$) in (4.14) is the same as considering (4.14) in the orthogonal basis formed by the eigenvectors of $\mathbf{X}'\mathbf{X}$. Therefore, showing that (4.13) holds is equivalent to proving that

$$(\mathbf{V}\mathbf{z})'\mathbf{V}\mathbf{z} \geq (\mathbf{V}\mathbf{z})'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}\mathbf{V}\mathbf{z} \ .$$

This reduces to showing that $\mathbf{z}'\mathbf{z} \geq \mathbf{z}'\mathbf{\Lambda}(\mathbf{\Lambda} + a\mathbf{I})^{-1}\mathbf{z}$. Since $\mathbf{X}'\mathbf{X}$ is positive semi-definite all its eigenvalues are nonnegative. Therefore all the elements in the diagonal matrix $\mathbf{\Lambda}(\mathbf{\Lambda} + a\mathbf{I})^{-1}$ are $0 \leq \frac{\lambda_i}{\lambda_i + a} < 1$. It follows that $\mathbf{z}'\mathbf{z} > \mathbf{z}'\mathbf{\Lambda}(\mathbf{\Lambda} + a\mathbf{I})^{-1}\mathbf{z}$, which means that

$$\mathbf{x}'\mathbf{x} > \mathbf{x}'\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}\mathbf{x} \ . \tag{4.15}$$

We have just proved the linear case. The general kernel case can be obtained by using finite dimensional approximations. Recall that inherent in every kernel is a function ϕ that maps objects to the RKHS $\mathcal{F}$, which is isomorphic to $l_2 = \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots) | \sum_{i=1}^{\infty} \alpha_i^2 \text{ converges}\}$. Let us consider the sequence on subspaces $R_1 \subseteq R_2 \subseteq \ldots \subseteq \mathcal{F}$. The set $R_s = \{(\alpha_1, \alpha_2, \dots, \alpha_s, 0, 0, \dots)\}$ may be identified with $\mathbb{R}^s$. Let $p_s : \mathcal{F} \mapsto R_s$ be the projection operator $p_s(\boldsymbol{\alpha}) = (\alpha_1, \alpha_2, \dots, \alpha_s, 0, 0, \dots)$, $\phi_s : X \mapsto R_s$ be $\phi_s = p_s(\phi)$, and k_s be given by $k_s(\mathbf{v}_1, \mathbf{v}_2) = \langle \phi_s(\mathbf{v}_1), \phi_s(\mathbf{v}_2) \rangle$, where $\mathbf{v}_1, \mathbf{v}_2 \in X$.

Inequality (4.15) holds for k_s since R_s has a finite dimension. If (4.15) is violated, then its counterpart with some large s is violated too. Therefore, it follows that

$$k(\mathbf{x}, \mathbf{x}) \geq \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k} \ . \tag{4.16}$$

Notice that we do not have a strict inequality anymore since in the limit (4.16) may turn into an equality.

We have just shown that $0 \leq s < 1$, therefore $\gamma^{(m)}$ converges to some

point r. We will now analyse the last term of (4.9) (that is c) and consequently show that the point r coincides with the prediction made by KRR for the same signal. In the definition of c (see (4.12)) the term $\mathbf{y}'(\mathbf{K}+a\mathbf{I})^{-1}\mathbf{k}$ is in fact KRR's prediction, therefore $c = \gamma_{\text{KRR}}(1-s)$. This means that (4.9) can be rewritten as

$$\gamma^{(m)} = s\gamma^{(m-1)} + \gamma_{\text{KRR}}(1-s) \ . \tag{4.17}$$

At fixed point r we have $r = sr + \gamma_{\text{KRR}}(1-s)$, implying that $r = \gamma_{\text{KRR}}$, which ends our proof. ■

Remark 1 As it currently stands, to compute the IKAAR prediction for an iteration m it is necessary to compute all the previous ones. We will now show how any prediction can be computed directly. Given (4.17) and the fact that $\gamma^{(0)} = 0$, we will prove by induction that for all $m \geq 1$

$$\begin{aligned} \gamma^{(m)} &= \gamma_{\text{KRR}} - s^m\gamma_{\text{KRR}} \\ &= (1-s^m)\gamma_{\text{KRR}} \ . \end{aligned} \tag{4.18}$$

Recall that s is given by (4.11). Clearly, (4.18) holds for $m = 1$ since

$$\begin{aligned} \gamma^{(1)} &= s\gamma^{(1-1)} + (1-s)\gamma_{\text{KRR}} \\ &= s\gamma^{(0)} + \gamma_{\text{KRR}} - s\gamma_{\text{KRR}} \\ &= \gamma_{\text{KRR}} - s\gamma_{\text{KRR}} \ . \end{aligned}$$

Let us assume that (4.18) holds for any $m \geq 1$. We will now show that it also holds for $m+1$.

$$\begin{aligned} \gamma^{(m+1)} &= s\gamma^{(m)} + (1-s)\gamma_{\text{KRR}} \\ &= s\left(\gamma_{\text{KRR}} - s^m\gamma_{\text{KRR}}\right) + (1-s)\gamma_{\text{KRR}} \\ &= s\gamma_{\text{KRR}} - s^{m+1}\gamma_{\text{KRR}} + \gamma_{\text{KRR}} - s\gamma_{\text{KRR}} \\ &= \gamma_{\text{KRR}} - s^{m+1}\gamma_{\text{KRR}} \ . \end{aligned}$$

Since (4.18) holds for $m = 1$ and for $m+1$, then by the inductive principle it follows that it holds for any $m \geq 1$.

This formulation of IKAAR shows that the rate of convergence of its predictions to those of KRR is exponential and that there is no iterative procedure to be solved since it computes the prediction for any iteration directly. □

4.3.3 Controlled KAAR (CKAAR)

KAAR's predictions are so rigid because it tries to minimise the value of the predictions themselves. This is evident in the second term of its objective function (4.2) on page 45. In our new method, the Controlled Kernel Aggregating Algorithm for Regression (CKAAR), we try to control this behaviour by adding a coefficient to this second term such that at time T our objective is to minimise

$$\mathcal{L}_T(\text{CKAAR}) = a\|\mathbf{w}_{\text{C}}\|^2 + b\langle \mathbf{w}_{\text{C}}, \mathbf{x}_T\rangle^2 + \sum_{t=1}^{T-1}(y_t - \langle \mathbf{w}_{\text{C}}, \mathbf{x}_t\rangle)^2 \ , \tag{4.19}$$

where $a > 0$ and $b \geq 0$. It is immediately clear that when $b = 0$ CKAAR should behave exactly like KRR and conversely like KAAR when $b = 1$. When b is somewhere in between, CKAAR will output predictions that are not as rigid as those of KAAR and do not fluctuate as much as those of KRR, whereas when $b > 1$ CKAAR will provide even more regularisation than KAAR does.

Letting $\mathbf{w} = \mathbf{w}_{\text{C}}$ we can express (4.19) in matrix notation to give

$$\begin{aligned}\mathcal{L}_T(\text{CKAAR}) &= a(\mathbf{w}'\mathbf{w}) + b(\mathbf{w}'\mathbf{x}_T)^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})^2 \\ &= a(\mathbf{w}'\mathbf{w}) + \widetilde{\mathbf{y}}'\widetilde{\mathbf{y}} - 2\mathbf{w}'\widehat{\mathbf{X}}'\widetilde{\mathbf{y}} + \mathbf{w}'\widehat{\mathbf{X}}'\widehat{\mathbf{X}}\mathbf{w} \ ,\end{aligned}$$

where $\widehat{\mathbf{X}} = (\mathbf{X}', \sqrt{b}\,\mathbf{x}_T)'$ and $\widetilde{\mathbf{y}} = (\mathbf{y}', 0)'$. If we differentiate this with respect to $\mathbf{w}$, divide throughout by 2 and set it equal to 0 we get

$$\frac{1}{2}\frac{\partial \mathcal{L}_T(\text{CKAAR})}{\partial \mathbf{w}} = a\mathbf{w} - \widehat{\mathbf{X}}'\widetilde{\mathbf{y}} + \widehat{\mathbf{X}}'\widehat{\mathbf{X}}\mathbf{w} = 0 \ .$$

This means that the CKAAR solution ($\mathbf{w}_{\text{C}}$) to the regression problem for a

new example $\mathbf{x}_T$ is

$$\mathbf{w}_{\text{C}} = (a\mathbf{I} + \widehat{\mathbf{X}}'\widehat{\mathbf{X}})^{-1}\widehat{\mathbf{X}}'\widetilde{\mathbf{y}} \ .$$

The solution we have just derived is for the linear case only. To handle the nonlinear case we formulate our solution in dual variables using Lemma 4, so that we can apply the kernel trick:

$$\begin{aligned}
\gamma_{\text{CKAAR}} &= \mathbf{w}_{\text{C}}'\mathbf{x} \\
&= \left(\left(a\mathbf{I} + \widehat{\mathbf{X}}'\widehat{\mathbf{X}}\right)^{-1}\widehat{\mathbf{X}}'\widetilde{\mathbf{y}}\right)'\mathbf{x} \\
&= \widetilde{\mathbf{y}}'\left(a\mathbf{I} + \widehat{\mathbf{X}}\widehat{\mathbf{X}}'\right)^{-1}\widehat{\mathbf{X}}\mathbf{x} \ .
\end{aligned}$$

The kernel version of CKAAR makes a prediction for a new signal $\mathbf{x}_T$ by

$$\gamma_{\text{CKAAR}} = \widetilde{\mathbf{y}}'(a\mathbf{I} + \widehat{\mathbf{K}})^{-1}\widehat{\mathbf{k}} \ , \tag{4.20}$$

where

$$\widehat{\mathbf{K}} = \begin{bmatrix}
k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_{T-1}) & \sqrt{b}\,k(\mathbf{x}_1,\mathbf{x}_T) \\
k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdots & k(\mathbf{x}_2,\mathbf{x}_{T-1}) & \sqrt{b}\,k(\mathbf{x}_2,\mathbf{x}_T) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
k(\mathbf{x}_{T-1},\mathbf{x}_1) & k(\mathbf{x}_{T-1},\mathbf{x}_2) & \cdots & k(\mathbf{x}_{T-1},\mathbf{x}_{T-1}) & \sqrt{b}\,k(\mathbf{x}_{T-1},\mathbf{x}_T) \\
\sqrt{b}\,k(\mathbf{x}_T,\mathbf{x}_1) & \sqrt{b}\,k(\mathbf{x}_T,\mathbf{x}_2) & \cdots & \sqrt{b}\,k(\mathbf{x}_T,\mathbf{x}_{T-1}) & b\,k(\mathbf{x}_T,\mathbf{x}_T)
\end{bmatrix} ,$$

and

$$\widehat{\mathbf{k}} = \begin{bmatrix}
k(\mathbf{x}_1,\mathbf{x}_T) \\
k(\mathbf{x}_2,\mathbf{x}_T) \\
\vdots \\
k(\mathbf{x}_{T-1},\mathbf{x}_T) \\
\sqrt{b}\,k(\mathbf{x}_T,\mathbf{x}_T)
\end{bmatrix} .$$

Clearly, $(a\mathbf{I} + \widehat{\mathbf{K}})$ is still positive definite since $\widehat{\mathbf{K}}$ is a Gram matrix of vectors in Hilbert space and one of them happens to be multiplied by $\sqrt{b}$.

4.4 Summary of Methods and Comparisons with Ridge Regression

In this section we will be comparing our methods with KRR. First, we formulate our methods in terms of KRR's prediction γ_{KRR} and $z = k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}$. For IKAAR it suffices to notice that in (4.18) $s = z/(z + a)$. Therefore IKAAR's prediction can be written as

$$\gamma_{\text{IKAAR}} = \gamma_{\text{KRR}} \left(1 - \left(\frac{z}{z+a}\right)^m\right) \quad .$$

For CKAAR we will use Lemma 6 to get a similar formulation. From (4.20) we know that CKAAR's prediction for a new signal $\mathbf{x}_T$ is given by

$$\begin{aligned}
\gamma_{\text{CKAAR}} &= \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}' \begin{bmatrix} \mathbf{K} + a\mathbf{I} & \sqrt{b}\mathbf{k} \\ \sqrt{b}\mathbf{k}' & bk(\mathbf{x}_T, \mathbf{x}_T) + a \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{k} \\ \sqrt{b}k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}' \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{k} \\ \sqrt{b}k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}' \begin{bmatrix} \widetilde{\mathbf{P}} & \widetilde{\mathbf{Q}} \\ \widetilde{\mathbf{R}} & \widetilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \sqrt{b}k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} \\
&= \mathbf{y}'\widetilde{\mathbf{P}}\mathbf{k} + \sqrt{b}\mathbf{y}'\widetilde{\mathbf{Q}}k(\mathbf{x}_T, \mathbf{x}_T) \\
&= \mathbf{y}'\mathbf{P}^{-1}k + \mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}\mathbf{k} \\
&\quad - \sqrt{b}\mathbf{y}'\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}k(\mathbf{x}_T, \mathbf{x}_T) \\
&= \mathbf{y}'\mathbf{P}^{-1}\mathbf{k}\left(1 + (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}(\sqrt{b}\mathbf{R}\mathbf{P}^{-1}\mathbf{k} - bk(\mathbf{x}_T, \mathbf{x}_T))\right) \\
&= \gamma_{\text{KRR}}\left(1 - \frac{k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})\mathbf{k}}{k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})\mathbf{k} + a/b}\right) \\
&= \gamma_{\text{KRR}}\left(1 - \frac{z}{z + a/b}\right) \quad .
\end{aligned}$$

Since both IKAAR and CKAAR are generalisations of KAAR we can get a formulation for KAAR in terms of γ_{KRR} and z simply by considering the former

Table 4.1: Formulations in terms of γ_{KRR} and $z = k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}$.

KRR	$\gamma_{\text{KRR}} = \mathbf{y}'(a\mathbf{I} + \mathbf{K})^{-1}\mathbf{k}$	KAAR	$\gamma_{\text{KRR}}\left(1 - \frac{z}{z+a}\right)$
IKAAR	$\gamma_{\text{KRR}}\left(1 - \left(\frac{z}{z+a}\right)^m\right), m \geq 1$	CKAAR	$\gamma_{\text{KRR}}\left(1 - \frac{z}{z+a/b}\right), b \geq 0$
KOKO	$\gamma_{\text{KRR}}\left(1 - \theta\left(\frac{z}{z+a}\right)\right), 0 \leq \theta \leq 1$	KRRV	$\gamma_{\text{KRR}}\left(1 - v\right), 0 \leq v \leq 1$

with $m = 1$ or the latter with $b = 1$. This gives us

$$\gamma_{\text{KAAR}} = \gamma_{\text{KRR}}\left(1 - \frac{z}{z+a}\right) \quad .$$

Finally, we get KOKO's prediction in terms of KRR by a simple substitution of KAAR's prediction in (4.6) on page 51 which gives us

$$\gamma_{\text{KOKO}} = \gamma_{\text{KRR}}\left(1 - \theta\left(\frac{z}{z+a}\right)\right) \quad .$$

In Table 4.1 we present a summary of these formulations of the predictions made by KAAR, IKAAR, CKAAR and KOKO in terms of KRR's prediction γ_{KRR} and $z = k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}$. These formulations give us computational advantages and they also allow us to understand our new methods better. It is immediately clear that all these methods 'scale down' (in different ways) KRR's prediction towards 0 (recall that $a > 0$, that we have shown that $z \geq 0$, and that we assume that the mean of the outcomes is 0) in an effort to combat noise and outliers. Note that IKAAR, CKAAR and KOKO have a parameter which controls (or can completely remove) the extra regularisation introduced by KAAR. In the table we have also included another method, dubbed KRRV, to compare our methods against. KRRV simply scales down KRR's prediction using a scalar, whereas our methods take in consideration the signal for which the prediction is being made.

4.4.1 Bayesian Interpretation

In order to get a better understanding of the term $z = k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}$, we will now give a Bayesian derivation of Ridge Regression which mainly follows Melluish et al. [2001] and Vovk et al. [2005, Section 10.3] for the primary case. We will be giving our own derivation for the dual (kernel) case. This will provide us with the full predictive distribution for the new signal $\mathbf{x}_T$.

We are given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{T-1}, y_{T-1})$, where $\mathbf{x}_t \in X$ (here we take $X \subseteq \mathbb{R}^n$), and $y_t \in \mathbb{R}$. Let us assume that the signals are fixed (deterministic) and that the outcomes were generated by the linear model

$$y_t = \langle \mathbf{w}, \mathbf{x}_t \rangle + \varepsilon_t \ ,$$

(this is identical to (3.2) on page 38) where $\mathbf{w} \in \mathbb{R}^n$ is distributed as $N(0, \sigma^2/a\mathbf{I})$ and $\varepsilon_t \in \mathbb{R}$ is distributed as $N(0, \sigma^2)$, and that all of these random elements are independent.

We will first find $P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{T-1}, y_{T-1}))$ that is, the posterior density of $\mathbf{w}$ given the training data. Recall that the normal distribution's density is

$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{x-\mu}{2\sigma^2}\right) \ ,$$

where μ and σ^2 are the mean and the variance respectively. The multivariate form of this together with Bayes' rule give us

$$\begin{aligned}
&P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{T-1}, y_{T-1})) \\
&\quad \propto P(\mathbf{w}) \times P((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{T-1}, y_{T-1})|\mathbf{w}) \\
&\quad = \exp\left(-\frac{a}{2\sigma^2}\|\mathbf{w}\|^2\right) \prod_{t=1}^{T-1} \exp\left(-\frac{1}{2\sigma^2}(y_t - \langle \mathbf{w}, \mathbf{x}_t \rangle)^2\right) \\
&\quad = \exp\left(-\frac{1}{2\sigma^2}\left(a\|\mathbf{w}\|^2 + \sum_{t=1}^{T-1}(y_t - \langle \mathbf{w}, \mathbf{x}_t \rangle)^2\right)\right) \ . \qquad (4.21)
\end{aligned}$$

This attains its maximum at the $\mathbf{w}$ that minimises

$$a\|\mathbf{w}\|^2 + \sum_{t=1}^{T-1}(y_t - \langle \mathbf{w}, \mathbf{x}_t \rangle)^2 \ .$$

This is Ridge Regression's objective function (see (3.5) on page 39), therefore, the solution to this optimisation problem, which we shall denote by $\mathbf{w}_{\mathrm{R}}$, is (3.6) on page 39.

Rewriting (4.21) in matrix notation gives us

$$\begin{aligned} P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{T-1}, y_{T-1})) & \\ & \propto \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{w}'(\mathbf{X}'\mathbf{X} + a\mathbf{I})\mathbf{w} - 2\mathbf{y}'\mathbf{X}\mathbf{w} + \mathbf{y}'\mathbf{y}\right)\right) \\ & \propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{w} - \mathbf{w}_{\mathrm{R}})'(\mathbf{X}'\mathbf{X} + a\mathbf{I})(\mathbf{w} - \mathbf{w}_{\mathrm{R}})\right) \ . \end{aligned} \tag{4.22}$$

Equation (4.22) can be recognised as the probability distribution function of the multivariate normal distribution with mean $\mathbf{w}_{\mathrm{R}}$ and covariance matrix $\sigma^2(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}$.

Therefore, the predictive distribution of $\langle \mathbf{w}, \mathbf{x}_T \rangle$, given the training data, is

$$N\left(\mathbf{x}_T'(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}\mathbf{X}'\mathbf{y},\ \sigma^2\mathbf{x}_T'(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}\mathbf{x}_T\right) \ . \tag{4.23}$$

We have just found the mean (which is equal to the prediction) and the variance of a prediction made by Ridge Regression in linear primary form. We are, however, interested in the distribution of predictions made by the dual (kernel) form. We have already given in (3.7) on page 40 the mean of this formulation. We will derive the dual form of the variance by using Lemma 4.

Starting from the variance of the primary form we have

$$\begin{aligned}
\sigma^2 \mathbf{x}_T'(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1}\mathbf{x}_T &= \frac{\sigma^2}{a}\mathbf{x}_T'(a\mathbf{I}(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1})\mathbf{x}_T \\
&= \frac{\sigma^2}{a}\mathbf{x}_T'((\mathbf{X}'\mathbf{X} + a\mathbf{I} - \mathbf{X}'\mathbf{X})(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1})\mathbf{x}_T \\
&= \frac{\sigma^2}{a}\mathbf{x}_T'((\mathbf{X}'\mathbf{X} + a\mathbf{I})(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1} \\
&\quad - \ \mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + a\mathbf{I})^{-1})\mathbf{x}_T \\
&= \frac{\sigma^2}{a}\mathbf{x}_T'(\mathbf{I} - \mathbf{X}'(\mathbf{X}\mathbf{X}' + a\mathbf{I})^{-1}\mathbf{X})\mathbf{x}_T \\
&= \frac{\sigma^2}{a}\left(\mathbf{x}_T'\mathbf{x}_T - \mathbf{x}_T'\mathbf{X}'(\mathbf{X}\mathbf{X}' + a\mathbf{I})^{-1}\mathbf{X}\mathbf{x}_T\right) \ .
\end{aligned}$$

Therefore the distribution of Ridge Regression's prediction (4.23) in dual form is

$$N\left(\mathbf{y}'(\mathbf{X}\mathbf{X}' + a\mathbf{I})^{-1}\mathbf{X}\mathbf{x}_T,\ \frac{\sigma^2}{a}\left(\mathbf{x}_T'\mathbf{x}_T - \mathbf{x}_T'\mathbf{X}'(\mathbf{X}\mathbf{X}' + a\mathbf{I})^{-1}\mathbf{X}\mathbf{x}_T\right)\right) \ .$$

As usual, we now replace all dot products in the dual formulation with kernels to get the kernel version of this distribution (notice that the mean is equivalent to Kernel Ridge Regression's (KRR) prediction (3.8) on page 40)

$$N\left(\mathbf{y}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k},\ \frac{\sigma^2}{a}\left(k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}\right)\right) \ .$$

It is immediately clear that $z = k(\mathbf{x}_T, \mathbf{x}_T) - \mathbf{k}'(\mathbf{K} + a\mathbf{I})^{-1}\mathbf{k}$ is proportional to the variance of Ridge Regression's prediction. As seen above, KAAR, IKAAR, CKAAR and KOKO use z to decide on the amount by which to push the prediction towards the mean (IKAAR, CKAAR and KOKO have an extra parameter that controls the extent to which this happens). A large variance can be interpreted as a lack of confidence in the accuracy of the prediction. Pushing the prediction towards the mean of the outcomes by an amount proportional to this variance can be a way of preventing extra loss.

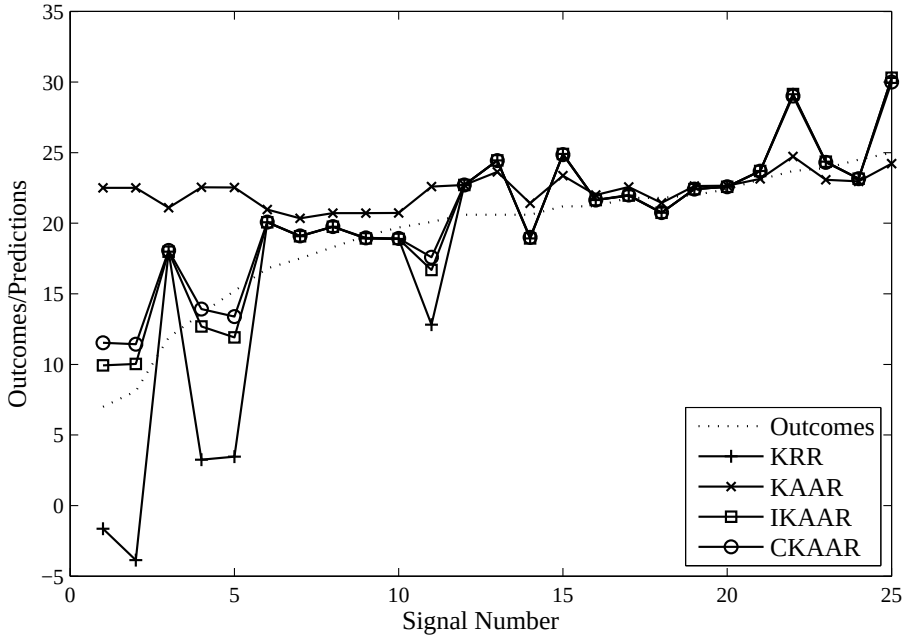


Figure 4.2: KRR, KAAR, IKAAR and CKAAR approximating a signal-outcome behaviour.

4.5 Empirical Results

In this section we present the empirical performance of KRR, KAAR and our new methods on two popular datasets, namely the Gaze dataset [Quiñonero-Candela et al., 2006] and the Boston Housing dataset [Asuncion and Newman, 2007]. The results of experiments on the artificial Mexican Hat dataset and on Abalone, Auto-MPG, Auto-Price, Relative CPU Performance, Servo and Wisconsin Prognostic Breast Cancer datasets (all from Asuncion and Newman [2007]) can be found in Appendix B. These other empirical results follow a similar pattern to those reported in this section.

We will first revisit the motivation for introducing changes to KAAR and show a new version of Figure 4.1 on page 50 with IKAAR and CKAAR predictions included. For these results, which are shown in Figure 4.2, the CKAAR control parameter $b = 0.01$ and IKAAR's iteration $m = 88$ (these parameters

were chosen manually to show the capabilities of these two methods). Using these parameters, both IKAAR and CKAAR approximate the true outcomes better than either KRR or KAAR, suffering a square loss of 7.44 and 7.51 respectively. In addition, the standard deviation of IKAAR's predictions is 5.17, whereas that of CKAAR is 4.68. This shows that these two methods can be made not to fluctuate as much as KRR while not being as rigid as KAAR. This behaviour was indeed our objective.

For all our experiments we used four standard kernels: polynomial, spline, ANOVA spline, and RBF (see Section 3.2.1). Note that in the results tables we include p-values denoting the statistical significance of the difference in the results of the methods. When no statistical significance is achieved the corresponding p-values are prefixed with an asterisk ($*$) (see below for an outline of statistical significance and p-values).

4.5.1 Experimentation Methodology

Let $d_1, d_2, \ldots, d_p$ be p random permutations of a dataset d. Let s_i be a set of parameters required by a method m. For example, in the case of Kernel Ridge Regression, this parameter set would consist of a value for a, the kernel (for instance the polynomial kernel), and any parameters needed by the kernel (for example the degree of the polynomial kernel). Given l different parameter sets $(s_1, s_2, \ldots, s_l)$, we apply the following procedure to measure the predictive performance of a method m on a dataset d. This procedure is commonly used in the machine learning community (see, for example, Drucker et al. [1997], Saunders et al. [1998], and Stitson et al. [1999]).

for $i = 1$ to p **do** {for every permutation of the dataset}
  {— here starts the validation stage —}
  Split d_i into 3 sequential parts: $d_i^{(1)}$ (training), $d_i^{(2)}$ (validation), and $d_i^{(3)}$ (testing). {the sizes of these are specified beforehand}
  for $j = 1$ to l **do** {for every parameter set}
    Train method m using parameters s_j on the training set $d_i^{(1)}$.
    Make predictions with m using s_j on the validation set $d_i^{(2)}$ in batch mode.

Calculate v_{ij} which is the mean loss suffered by m on $d_i^{(2)}$ using parameters s_j.

end for

$h = \arg\min_{j:1,\ldots,l} v_{ij}$ {h is the index of the parameter set which performed best during validation. If this is not unique we take the first one.}

{— here starts the testing stage —}

Train method m using parameters s_h on the training set $d_i^{(1)}$

Make predictions with m using s_h on the testing set $d_i^{(3)}$ in batch or online mode.

Calculate e_i which is the mean loss suffered on d_i in the testing stage.

end for

return $r = \frac{1}{p}\sum_{i=1}^{p} e_i$ {this is the average of the mean losses suffered on the testing sets of all permutations of the dataset d}

In the results tables we report this value r, which we shall call the Mean Square Error (MSE), and the standard deviation (SD) of the mean losses for all the methods mentioned in this paper.

4.5.2 Normalisation

It is considered good practice to normalise or standardise the data prior to applying an algorithm to it. Features that are too big can cause computational problems and a feature that is consistently much larger than another one may be given undue extra importance. Since the spline kernels require that all the features in the signals be nonnegative we chose to normalise the features to the interval $[0, 1]$. Let $\mathbf{X}$ be the matrix containing all the signals (ℓ in total) in the dataset, one per row. It follows that every column of $\mathbf{X}$ corresponds to all the values of a particular feature in the dataset. Let x_{ij} be the element at the ith row and the jth column of $\mathbf{X}$. Given a signal $\mathbf{z}$ of length n, the normalised version of its jth element z_j which we shall denote with $\overline{z}_j$, is calculated by

$$\overline{z}_j = \frac{z_j - m_j}{r_j} \ ,$$

where $m_j = \min_{i=1}^{\ell}(x_{ij})$ and $r_j = \left(\max_{i=1}^{\ell}(x_{ij}) - \min_{i=1}^{\ell}(x_{ij})\right)$. Therefore, the normalised version of $\mathbf{z}$ is $\overline{\mathbf{z}} = (\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_n)'$.

Recall that we assume that the outcomes have a mean of 0. This is beneficial to our methods since KAAR's upper bound on its loss features Y, which is the largest possible absolute value of the outcomes (see Section 4.2). Making 0 the mean of the outcomes means making Y smaller, giving us a better loss upper bound. In addition, all our methods push KRR's prediction towards 0 as seen in Section 4.4. We cannot use the outcomes in the testing set, since these have to be predicted, therefore, the translation of an outcome is done by subtracting from it the mean of the training outcomes. Clearly, every prediction made must then be shifted up by this same amount.

In our experiments we also normalise our kernels (see Shawe-Taylor and Cristianini [2005, Section 5.1]). Recall that a kernel function computes the dot product of the images under a mapping ϕ of two signals

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \ .$$

We want to use the following normalised transformation instead

$$\overline{\phi}(\mathbf{x}) = \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|} \ .$$

It is easy to see that the norm of $\phi(\mathbf{x})$ is given by

$$\begin{aligned} \|\phi(\mathbf{x})\| &= \sqrt{\|\phi(\mathbf{x})\|^2} \\ &= \sqrt{\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle} \\ &= \sqrt{k(\mathbf{x}, \mathbf{x})} \ . \end{aligned}$$

Therefore, the normalised kernel $\overline{k}$ is given by

$$\begin{aligned}\overline{k}(\mathbf{x},\mathbf{z}) &= \left\langle \overline{\phi}(\mathbf{x}), \overline{\phi}(\mathbf{z}) \right\rangle \\ &= \left\langle \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|}, \frac{\phi(\mathbf{z})}{\|\phi(\mathbf{z})\|} \right\rangle \\ &= \frac{\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle}{\|\phi(\mathbf{x})\| \|\phi(\mathbf{z})\|} \\ &= \frac{k(\mathbf{x},\mathbf{z})}{\sqrt{k(\mathbf{x},\mathbf{x}) k(\mathbf{z},\mathbf{z})}} \ .\end{aligned}$$

4.5.3 Statistical Significance

It is not always obvious whether the difference between two sets of results is really an improvement or not especially if the difference is small or the results have a large variance. For instance, could this difference be due to chance alone? There exist several statistical tests that output a p-value, which is the probability of obtaining a result at least as extreme as the one given due to chance alone if the null hypothesis is true. Statistical significance tests can be broadly split up into two classes: parametric and nonparametric. The first assume that the data follows a particular distribution (typically the normal distribution) while the latter do not make such assumptions. Since we do not know what the distribution of the differences between the results of two methods is[2], we use the Wilcoxon Signed Rank Test (WSRT) which is a popular nonparametric test (see, for example, Hollander and Wolfe [1973]).

The WSRT tests the null hypothesis that the median difference between two matched samples, $\mathbf{r}_1$ and $\mathbf{r}_2$, is zero. In our case, $\mathbf{r}_1$ and $\mathbf{r}_2$ are the losses suffered by two methods on permutations of the same data. More specifically, $\mathbf{r}_1 = \left(m_1^{(1)}, m_2^{(1)}, \ldots, m_p^{(1)}\right)$ and $\mathbf{r}_2 = \left(m_1^{(2)}, m_2^{(2)}, \ldots, m_p^{(2)}\right)$, where $m_i^{(j)}$ is the mean square loss suffered by method j on the ith permutation of the dataset. Take, for instance, the case where $m_i^{(1)} = m_i^{(2)}$ for all i except one, say, when $i = h$. If the difference $m_h^{(1)} - m_h^{(2)}$ is very large then the mean of $\mathbf{r}_1$ and $\mathbf{r}_2$ can be very different. This can lead us to believe that one of the methods

[2]Preliminary tests using the Kolmogorov-Smirnov test for normality (see Hollander and Wolfe [1973]) show that the differences between the results of two methods in our experiments are, in fact, not normally distributed.

is better than the other on this particular dataset. However, in this case the WSRT will output a large p-value, indicating that the difference can be due to chance only. On the other hand, if most of the losses in $\mathbf{r}_1$ are (slightly) larger than those in $\mathbf{r}_2$, then the p-value produced will be close to 0, indicating that the difference is in fact statistically significant, and therefore, the method that suffered the losses in $\mathbf{r}_2$ performs better than the other on this dataset. Note that by convention, the null hypothesis is rejected if the p-value obtained is less than 0.05 (or 5%).

In general it can be said that if a p-value is not small then this means that either the null hypothesis is true or that we do not have a sample that is big enough. In our case the sample size is just the number of experiments on random permutations of the dataset. On the other hand, if a p-value is small then it means that we have enough samples and we can refute the null-hypothesis, that is, we can be confident in our results.

4.5.4 The Gaze Dataset

The outcomes in the Gaze dataset [Quiñonero-Candela et al., 2006] are the horizontal positions of targets displayed on a computer monitor measured in pixels. The corresponding 12 features are measurements from head mounted cameras that focus on markers on the monitor and estimate the positions of the eyes of the subject looking at the monitor. Since the cameras occasionally lose their calibration, the dataset contains several severe outliers. Note that only the training and validation sets were used from the original dataset since the outcomes of the testing set were not available. We did not remove any of the signals for our experiments (not even the outliers), therefore, it is to be expected that the results have high variance.

The dataset used contains 450 examples and 1000 random permutations where taken (i.e., $p = 1000$). Each permutation of the dataset was split into 350, 70 and 30 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table 4.2 were used during the validation stage. See Table 4.3 and Table 4.4 for the results in online mode and batch mode respectively.

Table 4.2: Validation parameters used for experiments on the Gaze dataset.

Parameter Name	Values
Polynomial degree	$4, 5$
Spline	(no parameters)
ANOVA spline order	$2, 4, 6, 8, 10, 12$
RBF σ	$1, 4$
a	$2^{-15}, 2^{-13}, \ldots, 2^{-3}$
IKAAR m	$21, 41, \ldots, 161$
CKAAR b	$0, 0.01, 0.05, 0.1$
KOKO θ	$0, 0.01, 0.05, 0.1, 0.5$
KRRV v	$0, 0.01, 0.05, 0.1$

Table 4.3: Online mode results on 1000 random permutations of the Gaze dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**					
Poly	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	7.67	31.35		1×10^{-92}	1×10^{-16}	8×10^{-4}	7×10^{-3}	$*\ 3\times10^{-1}$
KAAR	3.54	0.94	1×10^{-92}		2×10^{-109}	7×10^{-96}	2×10^{-93}	1×10^{-93}
IKAAR	**3.03**	5.39	1×10^{-16}	2×10^{-109}		5×10^{-9}	2×10^{-12}	4×10^{-17}
CKAAR	5.88	25.59	8×10^{-4}	7×10^{-96}	5×10^{-9}		2×10^{-2}	2×10^{-3}
KOKO	8.00	32.34	7×10^{-3}	2×10^{-93}	2×10^{-12}	2×10^{-2}		1×10^{-4}
KRRV	7.92	32.38	$*\ 3\times10^{-1}$	1×10^{-93}	4×10^{-17}	2×10^{-3}	1×10^{-4}	
Spline	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	2.88	5.77		4×10^{-146}	$*\ 8\times10^{-2}$	$*\ 7\times10^{-1}$	$*\ 7\times10^{-1}$	$*\ 8\times10^{-1}$
KAAR	4.17	1.13	4×10^{-146}		3×10^{-156}	4×10^{-141}	2×10^{-142}	4×10^{-146}
IKAAR	**2.35**	2.35	$*\ 8\times10^{-2}$	3×10^{-156}		$*\ 1\times10^{-1}$	4×10^{-2}	3×10^{-2}
CKAAR	2.73	4.92	$*\ 7\times10^{-1}$	4×10^{-141}	$*\ 1\times10^{-1}$		$*\ 7\times10^{-1}$	$*\ 9\times10^{-1}$
KOKO	3.04	7.93	$*\ 7\times10^{-1}$	2×10^{-142}	4×10^{-2}	$*\ 7\times10^{-1}$		$*\ 6\times10^{-2}$
KRRV	2.92	6.02	$*\ 8\times10^{-1}$	4×10^{-146}	3×10^{-2}	$*\ 9\times10^{-1}$	$*\ 6\times10^{-2}$	
ANOVA	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	3.91	17.08		2×10^{-98}	3×10^{-7}	$*\ 2\times10^{-1}$	$*\ 9\times10^{-1}$	$*\ 9\times10^{-1}$
KAAR	3.28	0.91	2×10^{-98}		2×10^{-109}	4×10^{-99}	5×10^{-96}	6×10^{-97}
IKAAR	**2.58**	3.35	3×10^{-7}	2×10^{-109}		4×10^{-4}	4×10^{-8}	2×10^{-8}
CKAAR	3.16	6.51	$*\ 2\times10^{-1}$	4×10^{-99}	4×10^{-4}		$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$
KOKO	3.98	16.11	$*\ 9\times10^{-1}$	5×10^{-96}	4×10^{-8}	$*\ 2\times10^{-1}$		3×10^{-2}
KRRV	4.00	17.25	$*\ 9\times10^{-1}$	6×10^{-97}	2×10^{-8}	$*\ 2\times10^{-1}$	3×10^{-2}	
RBF	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	4.83	19.86		3×10^{-76}	2×10^{-13}	8×10^{-8}	2×10^{-2}	$*\ 2\times10^{-1}$
KAAR	2.91	0.77	3×10^{-76}		8×10^{-93}	5×10^{-81}	3×10^{-73}	2×10^{-75}
IKAAR	**2.49**	3.13	2×10^{-13}	8×10^{-93}		4×10^{-2}	1×10^{-10}	1×10^{-13}
CKAAR	4.49	19.53	8×10^{-8}	5×10^{-81}	4×10^{-2}		2×10^{-5}	7×10^{-8}
KOKO	4.99	20.15	2×10^{-2}	3×10^{-73}	1×10^{-10}	2×10^{-5}		4×10^{-5}
KRRV	4.79	19.69	$*\ 2\times10^{-1}$	2×10^{-75}	1×10^{-13}	7×10^{-8}	4×10^{-5}	

Table 4.4: Batch mode results on 1000 random permutations of the Gaze dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**					
Poly	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	7.69	31.21		6×10^{-91}	7×10^{-16}	4×10^{-3}	3×10^{-2}	$*\ 5\times10^{-1}$
KAAR	3.60	0.97	6×10^{-91}		8×10^{-113}	4×10^{-97}	5×10^{-92}	1×10^{-91}
IKAAR	**3.00**	5.19	7×10^{-16}	8×10^{-113}		3×10^{-9}	3×10^{-13}	2×10^{-16}
CKAAR	5.86	25.43	4×10^{-3}	4×10^{-97}	3×10^{-9}		3×10^{-2}	9×10^{-3}
KOKO	8.14	33.77	3×10^{-2}	5×10^{-92}	3×10^{-13}	3×10^{-2}		9×10^{-4}
KRRV	8.10	34.61	$*\ 5\times10^{-1}$	1×10^{-91}	2×10^{-16}	9×10^{-3}	9×10^{-4}	
Spline	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	2.96	6.80		2×10^{-148}	$*\ 3\times10^{-1}$	$*\ 5\times10^{-1}$	$*\ 8\times10^{-1}$	$*\ 8\times10^{-1}$
KAAR	4.26	1.18	2×10^{-148}		2×10^{-155}	6×10^{-143}	6×10^{-146}	6×10^{-149}
IKAAR	**2.37**	2.35	$*\ 3\times10^{-1}$	2×10^{-155}		$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$	5×10^{-2}
CKAAR	2.76	5.12	$*\ 5\times10^{-1}$	6×10^{-143}	$*\ 2\times10^{-1}$		$*\ 6\times10^{-1}$	$*\ 9\times10^{-1}$
KOKO	3.23	12.80	$*\ 8\times10^{-1}$	6×10^{-146}	$*\ 2\times10^{-1}$	$*\ 6\times10^{-1}$		$*\ 7\times10^{-2}$
KRRV	2.98	6.90	$*\ 8\times10^{-1}$	6×10^{-149}	5×10^{-2}	$*\ 9\times10^{-1}$	$*\ 7\times10^{-2}$	
ANOVA	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	4.42	32.31		1×10^{-98}	5×10^{-6}	$*\ 2\times10^{-1}$	$*\ 9\times10^{-1}$	$*\ 8\times10^{-1}$
KAAR	3.34	0.95	1×10^{-98}		4×10^{-109}	6×10^{-99}	3×10^{-96}	5×10^{-97}
IKAAR	**2.58**	3.27	5×10^{-6}	4×10^{-109}		4×10^{-4}	2×10^{-7}	2×10^{-8}
CKAAR	3.18	6.74	$*\ 2\times10^{-1}$	6×10^{-99}	4×10^{-4}		$*\ 2\times10^{-1}$	$*\ 1\times10^{-1}$
KOKO	4.44	29.66	$*\ 9\times10^{-1}$	3×10^{-96}	2×10^{-7}	$*\ 2\times10^{-1}$		3×10^{-2}
KRRV	4.51	32.41	$*\ 8\times10^{-1}$	5×10^{-97}	2×10^{-8}	$*\ 1\times10^{-1}$	3×10^{-2}	
RBF	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	4.82	19.66		8×10^{-78}	2×10^{-13}	1×10^{-7}	1×10^{-2}	$*\ 2\times10^{-1}$
KAAR	2.96	0.80	8×10^{-78}		4×10^{-95}	4×10^{-84}	5×10^{-75}	2×10^{-77}
IKAAR	**2.49**	3.08	2×10^{-13}	4×10^{-95}		2×10^{-2}	2×10^{-10}	2×10^{-13}
CKAAR	4.42	18.85	1×10^{-7}	4×10^{-84}	2×10^{-2}		2×10^{-5}	7×10^{-8}
KOKO	4.97	19.78	1×10^{-2}	5×10^{-75}	2×10^{-10}	2×10^{-5}		3×10^{-5}
KRRV	4.78	19.47	$*\ 2\times10^{-1}$	2×10^{-77}	2×10^{-13}	7×10^{-8}	3×10^{-5}	

Table 4.5: Validation parameters used for experiments on the Boston Housing dataset.

Parameter Name	Values
Polynomial degree	$4, 5$
Spline	(no parameters)
ANOVA spline order	$2, 4, 6, 8, 10, 13$
RBF σ	$0.25, 1, 4$
a	$2^{-15}, 2^{-13}, \ldots, 2^{-1}$
IKAAR m	$21, 41, \ldots, 161$
CKAAR b	$0, 0.01, 0.05, 0.1$
KOKO θ	$0, 0.01, 0.05, 0.1, 0.5$
KRRV v	$0, 0.01, 0.05, 0.1$

4.5.5 The Boston Housing Dataset

The Boston Housing dataset [Asuncion and Newman, 2007] concerns the prices of houses in the suburbs of Boston. A signal corresponds to a particular suburb and contains 13 attributes, including features like the amount of air pollution and the average number of rooms. An outcome is simply the median price of the houses in thousands of dollars.

The dataset contains 506 examples and each permutation of the dataset was split into 401, 80 and 25 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table 4.5 were used during the validation stage.

Table 4.6 and Table 4.7 respectively contain the online and batch mode results achieved on the 100 permutations of the dataset used in Saunders et al. [1998]. This means that these results are directly comparable to those reported there. On careful inspection though, it was noted that these permutations are not quite random and contain some common patterns. We therefore repeated the Boston Housing experiments on 1000 random permutations of our own. The online and batch mode results of these new experiments can be found in Table 4.8 and Table 4.9 respectively.

Table 4.6: Online mode results on the 100 permutations of the Boston Housing dataset from Saunders et al. [1998].

Method	**MSE**	**SD**	**Statistical Significance of Difference**					
Poly			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	9.19	5.08		2×10^{-12}	$*\ 9\times10^{-1}$	$*\ 9\times10^{-2}$	1×10^{-2}	$*\ 1\times10^{-1}$
KAAR	14.14	6.77	2×10^{-12}		3×10^{-13}	2×10^{-14}	7×10^{-14}	2×10^{-13}
IKAAR	9.09	5.10	$*\ 9\times10^{-1}$	3×10^{-13}		$*\ 2\times10^{-1}$	5×10^{-2}	$*\ 3\times10^{-1}$
CKAAR	**8.99**	5.03	$*\ 9\times10^{-2}$	2×10^{-14}	$*\ 2\times10^{-1}$		$*\ 5\times10^{-1}$	$*\ 8\times10^{-1}$
KOKO	9.00	4.83	1×10^{-2}	7×10^{-14}	5×10^{-2}	$*\ 5\times10^{-1}$		4×10^{-2}
KRRV	9.07	4.86	$*\ 1\times10^{-1}$	2×10^{-13}	$*\ 3\times10^{-1}$	$*\ 8\times10^{-1}$	4×10^{-2}	
Spline			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	7.48	3.69		5×10^{-22}	$*\ 4\times10^{-1}$	$*\ 3\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 4\times10^{-1}$
KAAR	14.36	7.11	5×10^{-22}		4×10^{-22}	3×10^{-23}	8×10^{-23}	1×10^{-22}
IKAAR	7.44	3.86	$*\ 4\times10^{-1}$	4×10^{-22}		$*\ 4\times10^{-1}$	$*\ 6\times10^{-1}$	$*\ 6\times10^{-1}$
CKAAR	**7.37**	3.69	$*\ 3\times10^{-1}$	3×10^{-23}	$*\ 4\times10^{-1}$		$*\ 8\times10^{-1}$	$*\ 7\times10^{-1}$
KOKO	7.43	3.64	$*\ 2\times10^{-1}$	8×10^{-23}	$*\ 6\times10^{-1}$	$*\ 8\times10^{-1}$		$*\ 4\times10^{-1}$
KRRV	7.46	3.65	$*\ 4\times10^{-1}$	1×10^{-22}	$*\ 6\times10^{-1}$	$*\ 7\times10^{-1}$	$*\ 4\times10^{-1}$	
ANOVA			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	7.49	3.28		4×10^{-16}	$*\ 9\times10^{-2}$	$*\ 3\times10^{-1}$	$*\ 5\times10^{-1}$	$*\ 9\times10^{-1}$
KAAR	11.43	4.88	4×10^{-16}		9×10^{-17}	2×10^{-17}	4×10^{-17}	2×10^{-16}
IKAAR	**7.36**	3.35	$*\ 9\times10^{-2}$	9×10^{-17}		$*\ 5\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 1\times10^{-1}$
CKAAR	7.37	3.31	$*\ 3\times10^{-1}$	2×10^{-17}	$*\ 5\times10^{-1}$		$*\ 3\times10^{-1}$	$*\ 3\times10^{-1}$
KOKO	7.44	3.23	$*\ 5\times10^{-1}$	4×10^{-17}	$*\ 2\times10^{-1}$	$*\ 3\times10^{-1}$		$*\ 2\times10^{-1}$
KRRV	7.50	3.21	$*\ 9\times10^{-1}$	2×10^{-16}	$*\ 1\times10^{-1}$	$*\ 3\times10^{-1}$	$*\ 2\times10^{-1}$	
RBF			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	8.23	3.86		4×10^{-16}	$*\ 4\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 5\times10^{-1}$
KAAR	12.41	5.44	4×10^{-16}		4×10^{-15}	6×10^{-16}	6×10^{-16}	7×10^{-16}
IKAAR	8.19	3.95	$*\ 4\times10^{-1}$	4×10^{-15}		$*\ 6\times10^{-1}$	$*\ 7\times10^{-1}$	$*\ 1\times10^{-}$
CKAAR	**8.17**	3.95	$*\ 2\times10^{-1}$	6×10^{-16}	$*\ 6\times10^{-1}$		$*\ 5\times10^{-1}$	$*\ 2\times10^{-1}$
KOKO	8.19	3.85	$*\ 2\times10^{-1}$	6×10^{-16}	$*\ 7\times10^{-1}$	$*\ 5\times10^{-1}$		$*\ 2\times10^{-1}$
KRRV	8.24	3.85	$*\ 5\times10^{-1}$	7×10^{-16}	$*\ 1\times10^{-}$	$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$	

Table 4.7: Batch mode results on the 100 permutations of the Boston Housing dataset from Saunders et al. [1998].

Method	**MSE**	**SD**	**Statistical Significance of Difference**					
Poly			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	9.47	5.48		2×10^{-12}	$*\ 7\times10^{-1}$	3×10^{-2}	6×10^{-3}	$*\ 1\times10^{-1}$
KAAR	14.55	7.37	2×10^{-12}		9×10^{-13}	4×10^{-14}	5×10^{-14}	1×10^{-13}
IKAAR	9.39	5.52	$*\ 7\times10^{-1}$	9×10^{-13}		$*\ 2\times10^{-1}$	$*\ 7\times10^{-2}$	$*\ 6\times10^{-1}$
CKAAR	9.28	5.44	3×10^{-2}	4×10^{-14}	$*\ 2\times10^{-1}$		$*\ 4\times10^{-1}$	$*\ 8\times10^{-1}$
KOKO	**9.27**	5.23	6×10^{-3}	5×10^{-14}	$*\ 7\times10^{-2}$	$*\ 4\times10^{-1}$		4×10^{-2}
KRRV	9.34	5.26	$*\ 1\times10^{-1}$	1×10^{-13}	$*\ 6\times10^{-1}$	$*\ 8\times10^{-1}$	4×10^{-2}	
Spline			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	7.72	4.10		3×10^{-22}	$*\ 5\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 5\times10^{-1}$
KAAR	14.81	7.80	3×10^{-22}		1×10^{-21}	2×10^{-23}	2×10^{-23}	6×10^{-23}
IKAAR	7.73	4.30	$*\ 5\times10^{-1}$	1×10^{-21}		$*\ 8\times10^{-1}$	$*\ 8\times10^{-1}$	$*\ 8\times10^{-1}$
CKAAR	**7.60**	4.11	$*\ 2\times10^{-1}$	2×10^{-23}	$*\ 8\times10^{-1}$		$*\ 7\times10^{-1}$	$*\ 4\times10^{-1}$
KOKO	7.66	4.03	$*\ 2\times10^{-1}$	2×10^{-23}	$*\ 8\times10^{-1}$	$*\ 7\times10^{-1}$		$*\ 5\times10^{-1}$
KRRV	7.69	4.04	$*\ 5\times10^{-1}$	6×10^{-23}	$*\ 8\times10^{-1}$	$*\ 4\times10^{-1}$	$*\ 5\times10^{-1}$	
ANOVA			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	7.73	3.60		4×10^{-16}	$*\ 2\times10^{-1}$	3×10^{-2}	$*\ 3\times10^{-1}$	$*\ 8\times10^{-1}$
KAAR	11.75	5.28	4×10^{-16}		5×10^{-17}	2×10^{-17}	3×10^{-17}	1×10^{-16}
IKAAR	7.60	3.69	$*\ 2\times10^{-1}$	5×10^{-17}		$*\ 9\times10^{-1}$	$*\ 6\times10^{-1}$	$*\ 2\times10^{-1}$
CKAAR	**7.59**	3.64	3×10^{-2}	2×10^{-17}	$*\ 9\times10^{-1}$		$*\ 3\times10^{-1}$	$*\ 2\times10^{-1}$
KOKO	7.67	3.53	$*\ 3\times10^{-1}$	3×10^{-17}	$*\ 6\times10^{-1}$	$*\ 3\times10^{-1}$		$*\ 2\times10^{-1}$
KRRV	7.72	3.53	$*\ 8\times10^{-1}$	1×10^{-16}	$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$	$*\ 2\times10^{-1}$	
RBF			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	8.55	4.44		3×10^{-15}	$*\ 3\times10^{-1}$	$*\ 3\times10^{-1}$	$*\ 1\times10^{-1}$	$*\ 5\times10^{-1}$
KAAR	12.72	5.87	3×10^{-15}		1×10^{-14}	3×10^{-15}	5×10^{-16}	2×10^{-15}
IKAAR	8.51	4.50	$*\ 3\times10^{-1}$	1×10^{-14}		$*\ 8\times10^{-1}$	$*\ 4\times10^{-1}$	$*\ 8\times10^{-1}$
CKAAR	8.49	4.53	$*\ 3\times10^{-1}$	3×10^{-15}	$*\ 8\times10^{-1}$		$*\ 6\times10^{-1}$	$*\ 3\times10^{-1}$
KOKO	**8.48**	4.40	$*\ 1\times10^{-1}$	5×10^{-16}	$*\ 4\times10^{-1}$	$*\ 6\times10^{-1}$		$*\ 2\times10^{-1}$
KRRV	8.54	4.42	$*\ 5\times10^{-1}$	2×10^{-15}	$*\ 8\times10^{-1}$	$*\ 3\times10^{-1}$	$*\ 2\times10^{-1}$	

Table 4.8: Online mode results on 1000 random permutations of the Boston Housing dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**					
Poly			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	10.76	7.88		2×10^{-117}	$*\ 6\times10^{-1}$	$*\ 9\times10^{-2}$	$*\ 2\times10^{-1}$	$*\ 5\times10^{-1}$
KAAR	16.90	10.93	2×10^{-117}		6×10^{-118}	2×10^{-126}	2×10^{-125}	1×10^{-120}
IKAAR	11.15	8.02	$*\ 6\times10^{-1}$	6×10^{-118}		$*\ 9\times10^{-1}$	2×10^{-2}	$*\ 8\times10^{-2}$
CKAAR	11.16	8.05	$*\ 9\times10^{-2}$	2×10^{-126}	$*\ 9\times10^{-1}$		2×10^{-2}	$*\ 1\times10^{-1}$
KOKO	10.77	7.78	$*\ 2\times10^{-1}$	2×10^{-125}	2×10^{-2}	2×10^{-2}		9×10^{-3}
KRRV	**10.75**	7.82	$*\ 5\times10^{-1}$	1×10^{-120}	$*\ 8\times10^{-2}$	$*\ 1\times10^{-1}$	9×10^{-3}	
Spline			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	**9.76**	6.81		8×10^{-150}	$*\ 6\times10^{-2}$	$*\ 3\times10^{-1}$	$*\ 6\times10^{-1}$	$*\ 9\times10^{-1}$
KAAR	16.49	10.52	8×10^{-150}		6×10^{-165}	2×10^{-166}	9×10^{-161}	8×10^{-157}
IKAAR	10.14	7.36	$*\ 6\times10^{-2}$	6×10^{-165}		$*\ 2\times10^{-1}$	$*\ 8\times10^{-1}$	$*\ 9\times10^{-1}$
CKAAR	10.13	7.21	$*\ 3\times10^{-1}$	2×10^{-166}	$*\ 2\times10^{-1}$		$*\ 7\times10^{-2}$	$*\ 4\times10^{-1}$
KOKO	9.82	6.84	$*\ 6\times10^{-1}$	9×10^{-161}	$*\ 8\times10^{-1}$	$*\ 7\times10^{-2}$		$*\ 6\times10^{-2}$
KRRV	9.79	6.87	$*\ 9\times10^{-1}$	8×10^{-157}	$*\ 9\times10^{-1}$	$*\ 4\times10^{-1}$	$*\ 6\times10^{-2}$	
ANOVA			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	10.15	7.12		2×10^{-121}	$*\ 5\times10^{-2}$	$*\ 8\times10^{-1}$	2×10^{-2}	$*\ 4\times10^{-1}$
KAAR	15.28	9.84	2×10^{-121}		1×10^{-128}	2×10^{-135}	4×10^{-130}	4×10^{-126}
IKAAR	10.42	7.48	$*\ 5\times10^{-2}$	1×10^{-128}		$*\ 6\times10^{-1}$	$*\ 5\times10^{-1}$	$*\ 4\times10^{-1}$
CKAAR	10.39	7.26	$*\ 8\times10^{-1}$	2×10^{-135}	$*\ 6\times10^{-1}$		$*\ 1\times10^{-1}$	$*\ 9\times10^{-1}$
KOKO	**10.11**	7.05	2×10^{-2}	4×10^{-130}	$*\ 5\times10^{-1}$	$*\ 1\times10^{-1}$		5×10^{-3}
KRRV	10.15	7.12	$*\ 4\times10^{-1}$	4×10^{-126}	$*\ 4\times10^{-1}$	$*\ 9\times10^{-1}$	5×10^{-3}	
RBF			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	10.56	7.53		2×10^{-105}	5×10^{-4}	$*\ 5\times10^{-1}$	2×10^{-2}	$*\ 9\times10^{-1}$
KAAR	15.25	9.58	2×10^{-105}		8×10^{-128}	2×10^{-127}	7×10^{-120}	3×10^{-113}
IKAAR	10.44	7.37	5×10^{-4}	8×10^{-128}		1×10^{-2}	$*\ 9\times10^{-1}$	$*\ 1\times10^{-1}$
CKAAR	10.54	7.24	$*\ 5\times10^{-1}$	2×10^{-127}	1×10^{-2}		$*\ 1\times10^{-1}$	$*\ 9\times10^{-1}$
KOKO	**10.38**	7.24	2×10^{-2}	7×10^{-120}	$*\ 9\times10^{-1}$	$*\ 1\times10^{-1}$		1×10^{-2}
KRRV	10.46	7.45	$*\ 9\times10^{-1}$	3×10^{-113}	$*\ 1\times10^{-1}$	$*\ 9\times10^{-1}$	1×10^{-2}	

Table 4.9: Batch mode results on 1000 random permutations of the Boston Housing dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**					
Poly			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	**10.91**	8.02		4×10^{-116}	$*\ 9\times10^{-1}$	$*\ 9\times10^{-2}$	$*\ 1\times10^{-1}$	$*\ 3\times10^{-1}$
KAAR	17.14	11.23	4×10^{-116}		7×10^{-116}	6×10^{-126}	5×10^{-125}	5×10^{-120}
IKAAR	11.32	8.29	$*\ 9\times10^{-1}$	7×10^{-116}		$*\ 9\times10^{-1}$	6×10^{-3}	$*\ 7\times10^{-2}$
CKAAR	11.33	8.29	$*\ 9\times10^{-2}$	6×10^{-126}	$*\ 9\times10^{-1}$		2×10^{-2}	$*\ 9\times10^{-2}$
KOKO	10.94	8.02	$*\ 1\times10^{-1}$	5×10^{-125}	6×10^{-3}	2×10^{-2}		1×10^{-2}
KRRV	10.92	8.06	$*\ 3\times10^{-1}$	5×10^{-120}	$*\ 7\times10^{-2}$	$*\ 9\times10^{-2}$	1×10^{-2}	
Spline			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	**9.95**	7.41		2×10^{-148}	$*\ 6\times10^{-2}$	$*\ 5\times10^{-1}$	$*\ 4\times10^{-1}$	$*\ 8\times10^{-1}$
KAAR	16.74	10.86	2×10^{-148}		7×10^{-161}	2×10^{-164}	9×10^{-159}	8×10^{-155}
IKAAR	10.28	7.54	$*\ 6\times10^{-2}$	7×10^{-161}		$*\ 4\times10^{-1}$	$*\ 8\times10^{-1}$	$*\ 9\times10^{-1}$
CKAAR	10.32	7.79	$*\ 5\times10^{-1}$	2×10^{-164}	$*\ 4\times10^{-1}$		$*\ 9\times10^{-2}$	$*\ 4\times10^{-1}$
KOKO	10.02	7.41	$*\ 4\times10^{-1}$	9×10^{-159}	$*\ 8\times10^{-1}$	$*\ 9\times10^{-2}$		$*\ 6\times10^{-2}$
KRRV	9.97	7.43	$*\ 8\times10^{-1}$	8×10^{-155}	$*\ 9\times10^{-1}$	$*\ 4\times10^{-1}$	$*\ 6\times10^{-2}$	
ANOVA			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	10.34	7.72		9×10^{-118}	$*\ 6\times10^{-2}$	$*\ 1\times10^{-}$	1×10^{-2}	$*\ 3\times10^{-1}$
KAAR	15.48	10.10	9×10^{-118}		2×10^{-123}	3×10^{-131}	6×10^{-126}	4×10^{-124}
IKAAR	10.57	7.67	$*\ 6\times10^{-2}$	2×10^{-123}		$*\ 7\times10^{-1}$	$*\ 3\times10^{-1}$	$*\ 7\times10^{-1}$
CKAAR	10.60	7.90	$*\ 1\times10^{-}$	3×10^{-131}	$*\ 7\times10^{-1}$		$*\ 1\times10^{-1}$	$*\ 9\times10^{-1}$
KOKO	**10.31**	7.64	1×10^{-2}	6×10^{-126}	$*\ 3\times10^{-1}$	$*\ 1\times10^{-1}$		7×10^{-3}
KRRV	10.33	7.69	$*\ 3\times10^{-1}$	4×10^{-124}	$*\ 7\times10^{-1}$	$*\ 9\times10^{-1}$	7×10^{-3}	
RBF			KRR	KAAR	IKAAR	CKAAR	KOKO	KRRV
KRR	10.79	8.22		2×10^{-101}	2×10^{-5}	$*\ 5\times10^{-1}$	8×10^{-3}	$*\ 5\times10^{-1}$
KAAR	15.45	9.86	2×10^{-101}		5×10^{-123}	2×10^{-122}	2×10^{-116}	2×10^{-109}
IKAAR	10.60	7.68	2×10^{-5}	5×10^{-123}		5×10^{-3}	$*\ 8\times10^{-1}$	$*\ 9\times10^{-2}$
CKAAR	10.78	7.98	$*\ 5\times10^{-1}$	2×10^{-122}	5×10^{-3}		$*\ 5\times10^{-2}$	$*\ 7\times10^{-1}$
KOKO	**10.59**	7.88	8×10^{-3}	2×10^{-116}	$*\ 8\times10^{-1}$	$*\ 5\times10^{-2}$		2×10^{-2}
KRRV	10.68	8.11	$*\ 5\times10^{-1}$	2×10^{-109}	$*\ 9\times10^{-2}$	$*\ 7\times10^{-1}$	2×10^{-2}	

Table 4.10: Percent improvements of our methods' results on those of KRR on the Gaze dataset in batch mode.

Method / Kernel	% Improvement on KRR			
	Poly	Spline	ANOVA	RBF
KAAR	53.18	−43.88	24.44	38.59
IKAAR	60.96	19.84	41.58	48.34
CKAAR	23.77	6.68	28.05	8.39
KOKO	−5.91	−9.27	−0.60	−3.08
KRRV	−5.41	−0.83	−2.10	0.82

4.5.6 Discussion

As shown in the Gaze dataset results tables, KAAR and our hybrid methods obtain a statistically significant improvement over Kernel Ridge Regression (KRR) in three cases (out of four kernels). In Table 4.10, we give the percentage improvement (i.e., how much less loss was suffered on average) per kernel of our methods' losses on those of KRR in the batch mode of learning. Here KAAR can be seen to benefit from its extra regularisation. This is also true for our hybrid methods IKAAR and CKAAR, with IKAAR suffering a loss that is always less than that of both KRR and KAAR. The 'control' methods KOKO and KRRV almost always suffer slightly more loss than KRR.

On more typical datasets our methods do not offer any advantages over Kernel Ridge Regression. Indeed, on the popular Boston Housing dataset, KAAR always suffers more loss than KRR. On the other hand, IKAAR and CKAAR give a slight improvement over KRR on the 100 dataset shuffles from Saunders et al. [1998]. However, this is not statistically significant and is not confirmed by more extensive experiments. Therefore, on the Boston Housing dataset IKAAR and CKAAR are comparable to KRR.

4.6 Conclusion

KAAR is the kernel version of an application of the Aggregating Algorithm to the problem of regression. It has very interesting theoretical properties; however, empirical analyses of KAAR show that its regularisation can be too

strong at times. To alleviate this we introduced several hybrid techniques that merge KAAR and Kernel Ridge Regression (KRR). Experiments show that our new methods can outperform KRR and KAAR on datasets that contain a lot of noise or severe outliers such as in the Gaze dataset. Moreover, on the popular Boston Housing dataset, which is more of a typical dataset, our hybrid methods perform comparably to KRR. We also gave a Bayesian interpretation where it can be seen that, under some assumptions, KAAR and our new methods push KRR's predictions towards the mean of the outcomes. The extent to which this happens depends on the variance of KRR's prediction itself.

Chapter 5

Regression with Changing Dependencies

Traditional methods in machine learning, like Ridge Regression, make significant assumptions about the data. The Aggregating Algorithm for Regression (AAR) does not make any such assumptions. However, it does assume that the relationship between signals and outcomes is fixed (see Section 4.2). In this chapter we are unwilling to make this assumption and consider the problem where the dependency of outcomes on signals can change with time. An example of where this may be true is in financial data, where prices and related parameters can change more or less continuously. We apply the Aggregating Algorithm to the pool of experts made up off all predictors that can change with time. This results in a new method, which we call the Kernel Aggregating Algorithm for Regression with Changing dependencies (KAARCh), that has good theoretical and empirical properties. For comparison, we also derive a simple method that is a weighted version of CKAAR introduced in Section 4.3.3.

5.1 Introduction

Consider the online regression problem where the dependence of the outcome y_t on the signal $\mathbf{x}_t$ can change with time. An example of this is the prediction of financial options implied volatility described in Section 5.4.2. Standard re-

gression techniques, like Ridge Regression, treat all training examples equally. In time series theory there is a method called Generalized Autoregressive Conditional Heteroskedasticity (GARCH) which assigns exponentially decreasing weights to old examples [Bollerslev, 1986] (see also Hull [2005, Chapter 19]). This method is used to estimate historical volatility in finance. We would like to extend this idea to the more general problem of online regression.

In Section 5.2 we present two methods as a solution to this problem: Weighted CKAAR (WeCKAAR) and the Kernel Aggregating Algorithm for Regression with Changing dependencies (KAARCh). WeCKAAR is a simple method that adds decaying weights to our hybrid method CKAAR from Section 4.3.3. KAARCh is a new method based on the Aggregating Algorithm (AA). The AA, described in Chapter 2, allows us to merge experts from large pools to obtain optimal strategies. Consider a sequence $\theta_1, \theta_2, \ldots$; let it make the prediction $(\theta_1 + \theta_2 + \ldots + \theta_t)' \mathbf{x}_t$ on trial t, where $\theta_1, \ldots, \theta_t, \mathbf{x}_t \in \mathbb{R}^n$. To get KAARCh, the AA is used to merge all predictors of this type. Clearly, our class of experts is very large and we cannot compete in a reasonable sense with every expert from this class. However, in Section 5.3 we show that KAARCh can perform almost as well as any predictor if the latter is not changing very rapidly, i.e., if each $\|\theta_t\|$ is small or only a few are nonzero. It turns out that WeCKAAR and KAARCh have very similar prediction formulae.

A similar problem is considered in Herbster and Warmuth [2001], Kivinen et al. [2004], and Cavallanti et al. [2007] for classification and regression. In these publications, this problem is referred to as the non-stationary or shifting target problem and the corresponding bounds are called shifting bounds. The work by Herbster and Warmuth is closest to ours. However, their methods are based on Gradient Descent and therefore their bounds are of a different type. For instance, since our approach is based on the Aggregating Algorithm we get a coefficient for the term representing the cumulative loss of the experts equal to 1 (see Theorems 5 and 6), whereas those in the bounds of Herbster and Warmuth [2001, Theorems 14–16] are greater than 1.

In practice, KAARCh can be used to predict parameters that change slowly with time. KAARCh is more computationally expensive than the techniques described in Herbster and Warmuth [2001], with time and space complexities

that grow with time. This is not desirable in an algorithm designed for online learning; however, a practical implementation is described in Section 5.2.2. Essentially, KAARCh is made to 'forget' older examples that do not affect the prediction too much. We report the empirical performance of WeCKAAR and KAARCh in Section 5.4; first on an artificial dataset, and then on options implied volatility data. These results show that when dealing with changing dependencies, KAARCh is an improvement on standard and weighted regression techniques. In addition, the performance of WeCKAAR and KAARCh on options implied volatility data provided by the Russian Trading System Stock Exchange (RTSSE) is comparable to that of the specially designed proprietary technique currently being used.

5.2 Methods

We are interested in making predictions in online regression where the dependency of y_t on $\mathbf{x}_t$ can change with time. We present two solutions to this problem: a simple method named WeCKAAR and our new method KAARCh. It is interesting that the prediction formulae of these two methods are very similar.

5.2.1 WeCKAAR

Weighted CKAAR (WeCKAAR) is a simple modification of CKAAR (introduced in Section 4.3.3) that employs a decaying factor such that old examples are given less importance. The objective of WeCKAAR at time T is to find a $\mathbf{w}$ that minimises

$$\mathcal{L}_T(\mathrm{W}) = a\|\mathbf{w}\|^2 + b\langle \mathbf{w}, \mathbf{x}_T\rangle^2 + \sum_{t=1}^{T-1} d_t(y_t - \langle \mathbf{w}, \mathbf{x}_t\rangle)^2 \ , \tag{5.1}$$

where $a > 0$, $b \geq 0$, $y_t \in \mathbb{R}$, $\mathbf{w}, \mathbf{x}_t \in \mathbb{R}^n$, and $d_t \in \mathbb{R}$ are nonnegative weights that increase with t. Let $d_T = b$ and $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_T)$ be the diagonal

matrix with elements $d_1 \dots d_T$. Equation (5.1) can be rewritten as

$$\begin{aligned}\mathcal{L}_T(\mathrm{W}) &= a\|\mathbf{w}\|^2 + \left(\widetilde{\mathbf{y}} - \widetilde{\mathbf{X}}\mathbf{w}\right)' \mathbf{D} \left(\widetilde{\mathbf{y}} - \widetilde{\mathbf{X}}\mathbf{w}\right) \\ &= a\mathbf{w}'\mathbf{w} + \widetilde{\mathbf{y}}'\mathbf{D}\widetilde{\mathbf{y}} + \mathbf{w}'\widetilde{\mathbf{X}}'\mathbf{D}\widetilde{\mathbf{X}}\mathbf{w} - 2\mathbf{w}'\widetilde{\mathbf{X}}'\mathbf{D}\widetilde{\mathbf{y}} \ ,\end{aligned}$$

where $\widetilde{\mathbf{X}} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)'$ and $\widetilde{\mathbf{y}} = (y_1, y_2, \dots, y_{T-1}, 0)'$. If we differentiate this with respect to $\mathbf{w}$, divide throughout by 2 and make it equal to zero, we get

$$\frac{1}{2}\frac{\partial \mathcal{L}_T(\mathrm{W})}{\partial \mathbf{w}} = a\mathbf{w} + \widetilde{\mathbf{X}}'\mathbf{D}\widetilde{\mathbf{X}}\mathbf{w} - \widetilde{\mathbf{X}}'\mathbf{D}\widetilde{\mathbf{y}} = 0 \ .$$

This implies that

$$\mathbf{w} = \left(\widetilde{\mathbf{X}}'\mathbf{D}\widetilde{\mathbf{X}} + a\mathbf{I}\right)^{-1} \widetilde{\mathbf{X}}'\mathbf{D}\widetilde{\mathbf{y}} \ . \tag{5.2}$$

Dual (Kernel) Form

Using Lemma 4 we can obtain a form of WeCKAAR's prediction where signals appear only in dot products. Accordingly, a prediction for the signal $\mathbf{x}_T$ is made by

$$\begin{aligned}\gamma_T &= \mathbf{w}'\mathbf{x}_T \\ &= \widetilde{\mathbf{y}}'\sqrt{\mathbf{D}} \left(\sqrt{\mathbf{D}}\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}'\sqrt{\mathbf{D}} + a\mathbf{I}\right)^{-1} \sqrt{\mathbf{D}}\widetilde{\mathbf{X}}\mathbf{x}_T \ ,\end{aligned}$$

where $\sqrt{\mathbf{D}} = \mathrm{diag}\left(\sqrt{d_1}, \dots, \sqrt{d_T}\right)$. We now apply the kernel trick to obtain the kernel version of WeCKAAR

$$\gamma_T = \widetilde{\mathbf{y}}'\sqrt{\mathbf{D}} \left(\sqrt{\mathbf{D}}\widetilde{\mathbf{K}}\sqrt{\mathbf{D}} + a\mathbf{I}\right)^{-1} \sqrt{\mathbf{D}}\widetilde{\mathbf{k}} \ , \tag{5.3}$$

where

$$\sqrt{\mathbf{D}}\widetilde{\mathbf{K}}\sqrt{\mathbf{D}} = \begin{bmatrix} d_1 k(\mathbf{x}_1, \mathbf{x}_1) & \sqrt{d_1 d_2}k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \sqrt{d_1 d_T}k(\mathbf{x}_1, \mathbf{x}_T) \\ \sqrt{d_2 d_1}k(\mathbf{x}_2, \mathbf{x}_1) & d_2 k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \sqrt{d_2 d_T}k(\mathbf{x}_2, \mathbf{x}_T) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{d_T d_1}k(\mathbf{x}_T, \mathbf{x}_1) & \sqrt{d_T d_2}k(\mathbf{x}_T, \mathbf{x}_2) & \cdots & d_T k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} \ ,$$

and

$$\sqrt{\mathbf{D}}\widetilde{\mathbf{k}} = \begin{bmatrix} \sqrt{d_1}k(\mathbf{x}_1, \mathbf{x}_T) \\ \sqrt{d_2}k(\mathbf{x}_2, \mathbf{x}_T) \\ \vdots \\ \sqrt{d_T}k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} .$$

5.2.2 KAARCh

For our second new method, we apply the Aggregating Algorithm (AA) to the regression problem where the experts can change with time. We call this method the Aggregating Algorithm for Regression with Changing dependencies (AARCh). Subsequently, we will kernelise this to get Kernel AARCh (KAARCh).

AARCh: Primal Form

The main idea behind AARCh is to apply the Aggregating Algorithm to the case where the pool of experts is made up of all linear predictors that can change independently with time. We assume that outcomes are bounded by Y, $Y \in \mathbb{R}$, i.e., for any t, $y_t \in [-Y, Y]$ (we do not require our algorithm to know Y). We are interested in the square loss, therefore we will be using optimal $\eta = 1/(2Y^2)$ and substitution function (2.8) on page 31. For details on the square loss game, see Section 2.3.

An expert is a sequence $\theta_1, \theta_2, \ldots$, that at time T predicts

$$\mathbf{x}_T'(\theta_1 + \theta_2 + \ldots + \theta_T) \ ,$$

where for any t, $\theta_t \in \mathbb{R}^n$ and $\mathbf{x}_T \in \mathbb{R}^n$. To apply the AA to this problem we need to define a lower triangular block matrix $\mathbf{L}$, and θ which is a concatenation

of all the θ_t for $t = 1 \ldots T$, such that[1]

$$\mathbf{L}\theta = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{T-1} \\ \theta_T \end{bmatrix}$$

$$= \begin{bmatrix} \theta_1 \\ \theta_1 + \theta_2 \\ \vdots \\ \theta_1 + \theta_2 + \cdots + \theta_{T-1} \\ \theta_1 + \theta_2 + \cdots + \theta_{T-1} + \theta_T \end{bmatrix} .$$

The matrices $\mathbf{I}$ and $\mathbf{0}$ in $\mathbf{L}$ are the $n \times n$ identity and all-zero matrices respectively. We also need to define $\mathbf{z}_t$ which is $\mathbf{x}_t$ padded with zeros in the following way

$$\mathbf{z}_t = \begin{bmatrix} \underbrace{0 \quad \cdots \quad 0}_{n(t-1)} & \mathbf{x}_t' & \underbrace{0 \quad \cdots \quad 0}_{n(T-t)} \end{bmatrix}' ,$$

so that

$$\mathbf{z}_t' \mathbf{L} \theta = \mathbf{x}_t' (\theta_1 + \theta_2 + \ldots + \theta_t) \ .$$

Let $a_t > 0$, $t = 1, \ldots, T$, be arbitrary constants. Consider the prior distribution P_0 in the set $\mathbb{R}^{nT}$ of possible weights θ with the Gaussian density

$$P_0(d\theta) = \left(\prod_{t=1}^{T} a_t \right)^{n/2} \left(\frac{\eta}{\pi} \right)^{nT/2} e^{-\eta \sum_{t=1}^{T} a_t \|\theta_t\|^2} d\theta_1 \ldots d\theta_T$$

$$= \left(\left(\frac{\eta}{\pi} \right)^{T} \prod_{t=1}^{T} a_t \right)^{n/2} e^{-\eta \theta' \mathbf{A} \theta} d\theta \ ,$$

[1]The sum $\theta_1 + \ldots + \theta_t$ corresponds to the predictor $\mathbf{u}_t$ in Herbster and Warmuth [2001].

where, letting $\mathbf{I}$ and $\mathbf{0}$ be as above, we have

$$\mathbf{A} = \begin{bmatrix} a_1\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & a_2\mathbf{I} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & a_T\mathbf{I} \end{bmatrix} .$$

The loss of θ over the first T trials is

$$\begin{aligned} \mathrm{Loss}_T(\theta) &= \sum_{t=1}^{T} \left(y_t - \mathbf{z}_t'\mathbf{L}\theta\right)^2 \\ &= \theta'\mathbf{L}' \left(\sum_{t=1}^{T} \mathbf{z}_t\mathbf{z}_t'\right) \mathbf{L}\theta - 2\left(\sum_{t=1}^{T} y_t\mathbf{z}_t'\right)\mathbf{L}\theta + \sum_{t=1}^{T} y_t^2 \ . \end{aligned}$$

Therefore, the loss of the APA is (recall that $\beta = e^{-\eta}$)

$$\begin{aligned} \mathrm{Loss}_T(\mathrm{APA}) &= \log_\beta \int_{\mathbb{R}^{nT}} \beta^{\mathrm{Loss}_T(\theta)} P_0(d\theta) \\ &= \log_\beta \int_{\mathbb{R}^{nT}} \left(\left(\frac{\eta}{\pi}\right)^T \prod_{t=1}^{T} a_t\right)^{n/2} \\ &\qquad \times \ e^{-\eta\left(\theta'\mathbf{L}'\left(\sum_{t=1}^{T}\mathbf{z}_t\mathbf{z}_t'\right)\mathbf{L}\theta - 2\left(\sum_{t=1}^{T} y_t\mathbf{z}_t'\right)\mathbf{L}\theta + \sum_{t=1}^{T} y_t^2 + \theta'\mathbf{A}\theta\right)} d\theta \\ &= \log_\beta \int_{\mathbb{R}^{nT}} \left(\left(\frac{\eta}{\pi}\right)^T \prod_{t=1}^{T} a_t\right)^{n/2} \\ &\qquad \times e^{-\eta\theta'\left(\mathbf{L}'\sum_{t=1}^{T}\mathbf{z}_t\mathbf{z}_t'\mathbf{L} + \mathbf{A}\right)\theta + 2\eta\left(\sum_{t=1}^{T} y_t\mathbf{z}_t'\right)\mathbf{L}\theta - \eta\sum_{t=1}^{T} y_t^2} d\theta \ . \end{aligned}$$

Given the generalised prediction $g_T(\omega)$ which is the APA's loss with variable $\omega \in \mathbb{R}$ replacing y_T and using substitution function (2.8) on page 31, the AA's prediction is

$$\begin{aligned} \gamma_T &= \frac{1}{4Y} \log_\beta \frac{\beta^{g_T(-Y)}}{\beta^{g_T(Y)}} \\ &= \frac{1}{4Y} \log_\beta \frac{\int_{\mathbb{R}^{nT}} e^{-\eta\theta'\left(\mathbf{L}'\sum_{t=1}^{T}\mathbf{z}_t\mathbf{z}_t'\mathbf{L} + \mathbf{A}\right)\theta + 2\eta\left(\sum_{t=1}^{T-1} y_t\mathbf{z}_t'\mathbf{L} - Y\mathbf{z}_T'\mathbf{L}\right)\theta} d\theta}{\int_{\mathbb{R}^{nT}} e^{-\eta\theta'\left(\mathbf{L}'\sum_{t=1}^{T}\mathbf{z}_t\mathbf{z}_t'\mathbf{L} + \mathbf{A}\right)\theta + 2\eta\left(\sum_{t=1}^{T-1} y_t\mathbf{z}_t'\mathbf{L} + Y\mathbf{z}_T'\mathbf{L}\right)\theta} d\theta} \ . \end{aligned}$$

Let

$$Q_1(\theta) = \theta' \left(\mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \mathbf{A} \right) \theta - 2 \left(\sum_{t=1}^{T-1} y_t \mathbf{z}_t' \mathbf{L} - Y \mathbf{z}_T' \mathbf{L} \right) \theta \ , \text{ and}$$
$$Q_2(\theta) = \theta' \left(\mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \mathbf{A} \right) \theta - 2 \left(\sum_{t=1}^{T-1} y_t \mathbf{z}_t' \mathbf{L} + Y \mathbf{z}_T' \mathbf{L} \right) \theta \ .$$

By Lemma 8

$$\begin{aligned} \gamma_T &= \frac{1}{4Y} \log_\beta \frac{e^{-\eta \min_{\theta \in \mathbb{R}^{nT}} Q_1(\theta)}}{e^{-\eta \min_{\theta \in \mathbb{R}^{nT}} Q_2(\theta)}} \\ &= \frac{1}{4Y} \left(\min_{\theta \in \mathbb{R}^{nT}} Q_1(\theta) - \min_{\theta \in \mathbb{R}^{nT}} Q_2(\theta) \right) \ . \end{aligned}$$

Finally, by using Lemma 9 we get

$$\begin{aligned} \gamma_T &= \frac{1}{4Y} F \left(\mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \mathbf{A},\ -2 \sum_{t=1}^{T-1} y_t \mathbf{z}_t' \mathbf{L},\ 2Y \mathbf{z}_T' \mathbf{L} \right) \\ &= \left(\sum_{t=1}^{T-1} y_t \mathbf{z}_t' \right) \mathbf{L} \left(\mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \mathbf{A} \right)^{-1} \mathbf{L}' \mathbf{z}_T \ . \end{aligned} \tag{5.4}$$

AARCh: Dual Form

Let us define

$$\widetilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{z}_1' \\ \mathbf{z}_2' \\ \vdots \\ \mathbf{z}_T' \end{bmatrix} \ , \ \sqrt{\mathbf{A}} = \begin{bmatrix} \sqrt{a_1}\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sqrt{a_2}\mathbf{I} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \sqrt{a_T}\mathbf{I} \end{bmatrix} \ , \text{ and } \widetilde{\mathbf{y}} = \begin{bmatrix} y_1 \\ \vdots \\ y_{T-1} \\ 0 \end{bmatrix} \ .$$

We can rewrite (5.4) in matrix notation to get

$$\begin{aligned} \gamma_T &= \widetilde{\mathbf{y}}' \widetilde{\mathbf{Z}} \mathbf{L} \left(\mathbf{L}' \widetilde{\mathbf{Z}}' \widetilde{\mathbf{Z}} \mathbf{L} + \mathbf{A} \right)^{-1} \mathbf{L}' \mathbf{z}_T \\ &= \widetilde{\mathbf{y}}' \widetilde{\mathbf{Z}} \mathbf{L} \left(\sqrt{\mathbf{A}} \left(\sqrt{\mathbf{A}}^{-1} \mathbf{L}' \widetilde{\mathbf{Z}}' \widetilde{\mathbf{Z}} \mathbf{L} \sqrt{\mathbf{A}}^{-1} + \mathbf{I} \right) \sqrt{\mathbf{A}} \right)^{-1} \mathbf{L}' \mathbf{z}_T \\ &= \widetilde{\mathbf{y}}' \widetilde{\mathbf{Z}} \mathbf{L} \sqrt{\mathbf{A}}^{-1} \left(\sqrt{\mathbf{A}}^{-1} \mathbf{L}' \widetilde{\mathbf{Z}}' \widetilde{\mathbf{Z}} \mathbf{L} \sqrt{\mathbf{A}}^{-1} + \mathbf{I} \right)^{-1} \sqrt{\mathbf{A}}^{-1} \mathbf{L}' \mathbf{z}_T \ . \end{aligned}$$

We can now get a dual formulation of this by using Lemma 4:

$$\gamma_T = \widetilde{\mathbf{y}}' \left(\widetilde{\mathbf{Z}} \mathbf{L} \mathbf{A}^{-1} \mathbf{L}' \widetilde{\mathbf{Z}}' + \mathbf{I} \right)^{-1} \widetilde{\mathbf{Z}} \mathbf{L} \mathbf{A}^{-1} \mathbf{L}' \mathbf{z}_T \ . \tag{5.5}$$

KAARCh

Since in (5.5) signals appear only in dot products, we can use the kernel trick to introduce nonlinearity. In this case we get Kernel AARCh (KAARCh) that at time T makes a prediction

$$\gamma_T = \widetilde{\mathbf{y}}' \left(\bar{\mathbf{K}} + \mathbf{I} \right)^{-1} \bar{\mathbf{k}} \ , \tag{5.6}$$

where $\bar{\mathbf{K}} = \left(\left(\sum_{t=1}^{\min(i,j)} \frac{1}{a_t} \right) k(\mathbf{x}_i, \mathbf{x}_j) \right)_{i,j}$, for $i, j = 1, \ldots, T$, i.e.

$$\bar{\mathbf{K}} = \begin{bmatrix} \frac{1}{a_1} k(\mathbf{x}_1, \mathbf{x}_1) & \frac{1}{a_1} k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \frac{1}{a_1} k(\mathbf{x}_1, \mathbf{x}_T) \\ \frac{1}{a_1} k(\mathbf{x}_2, \mathbf{x}_1) & \left(\frac{1}{a_1} + \frac{1}{a_2} \right) k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \left(\frac{1}{a_1} + \frac{1}{a_2} \right) k(\mathbf{x}_2, \mathbf{x}_T) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{a_1} k(\mathbf{x}_T, \mathbf{x}_1) & \left(\frac{1}{a_1} + \frac{1}{a_2} \right) k(\mathbf{x}_T, \mathbf{x}_2) & \cdots & \left(\frac{1}{a_1} + \ldots + \frac{1}{a_T} \right) k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} ,$$

and $\bar{\mathbf{k}} = \left(\left(\sum_{t=1}^{i} \frac{1}{a_t} \right) k(\mathbf{x}_i, \mathbf{x}_T) \right)_i$, for $i = 1, \ldots, T$, i.e.

$$\bar{\mathbf{k}} = \begin{bmatrix} \frac{1}{a_1} k(\mathbf{x}_1, \mathbf{x}_T) \\ \left(\frac{1}{a_1} + \frac{1}{a_2} \right) k(\mathbf{x}_2, \mathbf{x}_T) \\ \vdots \\ \left(\frac{1}{a_1} + \ldots + \frac{1}{a_T} \right) k(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} .$$

Implementation Notes

For simplicity, we may take all equal $a_1, a_2, \ldots, a_T = a$. In this case, KAARCh's prediction formula (5.6) becomes

$$\gamma_T = \widetilde{\mathbf{y}}' \left(\breve{\mathbf{K}} + a\mathbf{I} \right)^{-1} \breve{\mathbf{k}} \ , \tag{5.7}$$

where

$$\breve{\mathbf{K}} = \begin{bmatrix} 1k(\mathbf{x}_1, \mathbf{x}_1) & 1k(\mathbf{x}_1, \mathbf{x}_2) & 1k(\mathbf{x}_1, \mathbf{x}_3) & \cdots & 1k(\mathbf{x}_1, \mathbf{x}_T) \\ 1k(\mathbf{x}_2, \mathbf{x}_1) & 2k(\mathbf{x}_2, \mathbf{x}_2) & 2k(\mathbf{x}_2, \mathbf{x}_3) & \cdots & 2k(\mathbf{x}_2, \mathbf{x}_T) \\ 1k(\mathbf{x}_3, \mathbf{x}_1) & 2k(\mathbf{x}_3, \mathbf{x}_2) & 3k(\mathbf{x}_3, \mathbf{x}_3) & \cdots & 3k(\mathbf{x}_3, \mathbf{x}_T) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1k(\mathbf{x}_T, \mathbf{x}_1) & 2k(\mathbf{x}_T, \mathbf{x}_2) & 3k(\mathbf{x}_T, \mathbf{x}_3) & \cdots & Tk(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} ,$$

and

$$\breve{\mathbf{k}} = \begin{bmatrix} 1k(\mathbf{x}_1, \mathbf{x}_T) \\ 2k(\mathbf{x}_2, \mathbf{x}_T) \\ 3k(\mathbf{x}_3, \mathbf{x}_T) \\ \vdots \\ Tk(\mathbf{x}_T, \mathbf{x}_T) \end{bmatrix} .$$

Recalling that a scalar multiplied by a kernel is still a kernel, and making allowances such that steps in time can be skipped (for instance there is no data available for some steps), the coefficients $1, \ldots, T$ in $\breve{\mathbf{K}}$ and $\breve{\mathbf{k}}$ can be replaced with any increasing, positive real numbers $t_1, \ldots, t_T$, representing the real world time at which examples arrive.

5.3 Upper Bounds

In this section we use the Aggregating Algorithm's properties to derive upper bounds on the cumulative square loss suffered by AARCh and KAARCh, compared to that of any expert in the pool.

5.3.1 AARCh Loss Upper Bound

Theorem 5 *For any point in time T and any $a_t > 0$, $t = 1, \ldots, T$,*

$$\mathrm{Loss}_T(AARCh) \le \inf_\theta \left(\mathrm{Loss}_T(\theta) + \sum_{t=1}^{T} a_t \|\theta_t\|^2 \right) + Y^2 \ln \det \left(\sqrt{\mathbf{A}}^{-1} \mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} \sqrt{\mathbf{A}}^{-1} + \mathbf{I} \right) \quad . \quad (5.8)$$

□

PROOF Given the Aggregating Algorithm's properties, we know that

$$\begin{aligned} \mathrm{Loss}_T(\mathrm{AARCh}) &\le \log_\beta \int_{\mathbb{R}^{nT}} \beta^{\mathrm{Loss}_T(\theta)} P_0(d\theta) \\ &= \log_\beta \left(\left(\frac{\eta}{\pi} \right)^T \prod_{t=1}^{T} a_t \right)^{n/2} \\ &\quad \times \int_{\mathbb{R}^{nT}} e^{-\eta \left(\theta' \left(\mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \mathbf{A} \right) \theta - 2 \left(\sum_{t=1}^{T} y_t \mathbf{z}_t \right) \mathbf{L} \theta + \sum_{t=1}^{T} y_t^2 \right)} d\theta \quad . \end{aligned}$$

By Lemma 8 this is equal to

$$
\begin{aligned}
&\inf_{\theta} \left(\mathrm{Loss}_T(\theta) + \theta' \mathbf{A} \theta\right) \\
&\qquad + \log_\beta \left(\left(\left(\frac{\eta}{\pi}\right)^T \prod_{t=1}^{T} a_t \right)^{n/2} \frac{\pi^{nT/2}}{\sqrt{\det\left(\eta \mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \eta \mathbf{A}\right)}} \right) \\
&= \inf_{\theta} \left(\mathrm{Loss}_T(\theta) + \sum_{t=1}^{T} a_t \|\theta_t\|^2 \right) + \log_\beta \sqrt{\frac{\left(\eta^T \prod_{t=1}^{T} a_t\right)^n}{\det\left(\eta \mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \eta \mathbf{A}\right)}} \\
&= \inf_{\theta} \left(\mathrm{Loss}_T(\theta) + \sum_{t=1}^{T} a_t \|\theta_t\|^2 \right) + \frac{1}{2} \log_\beta \left(\frac{\prod_{t=1}^{T} a_t^n}{\det\left(\mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} + \mathbf{A}\right)} \right) \\
&= \inf_{\theta} \left(\mathrm{Loss}_T(\theta) + \sum_{t=1}^{T} a_t \|\theta_t\|^2 \right) \\
&\qquad - \frac{1}{2} \log_\beta \left(\frac{\det\left(\sqrt{\mathbf{A}} \left(\sqrt{\mathbf{A}}^{-1} \mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} \sqrt{\mathbf{A}}^{-1} + \mathbf{I}\right) \sqrt{\mathbf{A}}\right)}{\prod_{t=1}^{T} a_t^n} \right) \\
&= \inf_{\theta} \left(\mathrm{Loss}_T(\theta) + \sum_{t=1}^{T} a_t \|\theta_t\|^2 \right) \\
&\qquad + Y^2 \ln \det \left(\sqrt{\mathbf{A}}^{-1} \mathbf{L}' \sum_{t=1}^{T} \mathbf{z}_t \mathbf{z}_t' \mathbf{L} \sqrt{\mathbf{A}}^{-1} + \mathbf{I} \right) \quad . \qquad \blacksquare
\end{aligned}
$$

5.3.2 KAARCh Loss Upper Bound

The following generalises Theorem 5. Note that we cannot repeat the proof for the linear case directly since it involves the evaluation of an integral over the space $\mathbb{R}^{nT}$.

Theorem 6 *Let k be a kernel on a space X, let D_t, $t = 1 \ldots T$, be any decision rules in the RKHS $\mathcal{F}$ induced by k and let $D = (D_1, D_2, \ldots, D_T)'$. Then, for*

any point in time T *and every* $a_t > 0$, $t = 1, \ldots, T$,

$$\mathrm{Loss}_T(KAARCh) \leq \mathrm{Loss}_T(D) + \sum_{t=1}^{T} a_t \|D_t\|^2 + Y^2 \ln\det\left(\bar{\mathbf{K}} + \mathbf{I}\right) \quad . \tag{5.9}$$

□

PROOF It will be sufficient to prove this for D_t of the form

$$f_t(\mathbf{x}) = \sum_{i=1}^{l^{(t)}} c_i^{(t)} k(\mathbf{v}_i^{(t)}, \mathbf{x}) \ ,$$

where $l^{(t)}$ are positive integers, $c_i^{(t)} \in \mathbb{R}$, and $\mathbf{v}_i^{(t)}, \mathbf{x} \in X$ (we use $^{(t)}$ to show that these parameters can be different for each f_t). This is because such finite sums are dense in the RKHS $\mathcal{F}$. If we take $f = (f_1, f_2, \ldots, f_T)'$, (5.9) becomes

$$\begin{aligned}\mathrm{Loss}_T(\mathrm{KAARCh}) &\leq \mathrm{Loss}_T(f) \\ &+ \sum_{t=1}^{T} a_t \sum_{i,j=1}^{l^{(t)}} c_i^{(t)} c_j^{(t)} k(\mathbf{v}_i^{(t)}, \mathbf{v}_j^{(t)}) + Y^2 \ln\det\left(\bar{\mathbf{K}} + \mathbf{I}\right) \quad , \end{aligned} \tag{5.10}$$

where

$$\mathrm{Loss}_T(f) = \sum_{t=1}^{T} \left(y_t - \sum_{i=1}^{l^{(t)}} c_i^{(t)} k(\mathbf{v}_i^{(t)}, \mathbf{x}_t) \right)^2 \quad .$$

In the special case when $X = \mathbb{R}^n$ and $k(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i' \mathbf{v}_j$ for every $\mathbf{v}_i, \mathbf{v}_j \in X$, (5.10) follows directly from (5.8). Indeed, a kernel predictor f_t reduces to the linear predictor $\theta_t = \sum_{i=1}^{l^{(t)}} c_i^{(t)} \mathbf{v}_i^{(t)}$ and the term $\sum_{i,j=1}^{l^{(t)}} c_i^{(t)} c_j^{(t)} k\left(\mathbf{v}_i^{(t)}, \mathbf{v}_j^{(t)}\right)$ equals the squared quadratic norm of θ_t. Finally, by Sylvester's determinant identity (see also Lemma 10 for an independent proof of this) we know that

$$\begin{aligned}\det\left(\bar{\mathbf{K}} + \mathbf{I}\right) &= \det\left(\widetilde{\mathbf{Z}}\mathbf{L}\mathbf{A}^{-1}\mathbf{L}'\widetilde{\mathbf{Z}}' + \mathbf{I}\right) \\ &= \det\left(\sqrt{\mathbf{A}}^{-1}\mathbf{L}'\widetilde{\mathbf{Z}}'\widetilde{\mathbf{Z}}\mathbf{L}\sqrt{\mathbf{A}}^{-1} + \mathbf{I}\right) \quad .\end{aligned}$$

The general case follows from the linear case in the limit because of finite-dimensional approximations (see the proof of Theorem 4 in Section 4.3.2). ■

5.3.3 Analysis

In this section we shall analyse upper bound (5.9) in order to obtain an equivalent of Corollary 1 in Section 4.2.1. Our goal is to show that KAARCh's cumulative loss is less or equal to that of a wide class of experts plus a term of the order $o(T)$.

Estimating the determinant of a positive definite matrix by the product of its diagonal elements (see Beckenbach and Bellman [1961, Section 2.10, Theorem 7]) and using the inequality $\ln(1+x) \leq x$ (in our case x is small, and therefore the resulting bound is tight), we get

$$\begin{aligned} Y^2 \ln\det\left(\bar{\mathbf{K}} + \mathbf{I}\right) &\leq Y^2 \sum_{t=1}^{T} \ln\left(1 + u^2 \sum_{i=1}^{t} \frac{1}{a_i}\right) \\ &\leq Y^2 u^2 \sum_{t=1}^{T} \sum_{i=1}^{t} \frac{1}{a_i} \\ &= Y^2 u^2 \sum_{t=1}^{T} \frac{T-t+1}{a_t} \quad , \end{aligned}$$

where $u = \sup_{\mathbf{x} \in X} \sqrt{k(\mathbf{x}, \mathbf{x})}$.

It is natural to single out the first decision rule D_1 and the corresponding coefficient a_1 from the rest. We may think of it as corresponding to the choice of the 'principal' dependency; let the rest of D_t ($t = 2, \ldots, T$) be small correction terms. Let us take equal $a_2 = \ldots = a_T = a$. The sum $\sum_{t=2}^{T}(T-t+1)/a_t$ becomes equal to $(1/a)\left((T-1) + (T-2) + \ldots + 1\right)$. This can be recognised as an arithmetic progression with $(T-1)$ terms and a common difference of 1. Therefore it becomes $T(T-1)/(2a)$ and we get

$$\begin{aligned} \mathrm{Loss}_T(\mathrm{KAARCh}) \leq \mathrm{Loss}_T(D) &+ \left(a_1 \|D_1\|^2 + \frac{Y^2 u^2 T}{a_1}\right) \\ &+ \left(a \sum_{t=2}^{T} \|D_t\|^2 + \frac{Y^2 u^2 T(T-1)}{2a}\right) \quad . \quad (5.11) \end{aligned}$$

If we bound the norm of D_1 by d_1 and assume that T is known in advance, a_1 may be chosen as in Corollary 1. The second term in the right hand side

of (5.11) can thus be bounded by $O\left(\sqrt{T}\right)$. If we assume that $\sum_{t=2}^{T} \|D_t\|^2 \leq s(T)$, where $s(T)$ is some function that depends on T, the last term of (5.11) becomes less or equal to

$$as(T) + \frac{Y^2u^2T(T-1)}{2a} \ . \tag{5.12}$$

If we differentiate (5.12) with respect to a and set this equal to zero we find the a that minimises the estimate above:

$$\begin{aligned} 0 &= s(T) - \frac{Y^2u^2T(T-1)}{2a^2} \\ \Longrightarrow \ a &= \sqrt{\frac{Y^2u^2T(T-1)}{2s(T)}} \ . \end{aligned}$$

This means that the third term in the right hand side of (5.11) can be bounded by $O\left(T\sqrt{s(T)}\right)$. If we substitute for a_1 and a in (5.11) with the terms we found that minimise it we get the following corollary:

Corollary 2 *Under the conditions of Theorem 6, let T be known in advance and $u = \sup_{\mathbf{x}\in X}\sqrt{k(\mathbf{x},\mathbf{x})}$. For every every $d_1 > 0$ and every function $s(T)$, if $\|D_1\| \leq d_1$ and $\sum_{t=2}^{T}\|D_t\|^2 \leq s(T)$, then a_t, for $t = 1,\ldots,T$, can be chosen so that*

$$\mathrm{Loss}_T(KAARCh) \leq \mathrm{Loss}_T(D) + 2Yud_1\sqrt{T} + 2Yu\sqrt{s(T)T(T-1)/2} \ . \tag{5.13}$$

If $s(T) = o(1)$, then $\mathrm{Loss}_T(KAARCh) \leq \mathrm{Loss}_T(D) + o(T)$. □

The estimate $s(T) = o(1)$ can be achieved in two natural ways. First, one can assume that each $\|D_t\|$, for $t = 2,\ldots,T$, is small:

$$\begin{aligned} \|D_t\|^2 &= \frac{1}{T-1}o(1) \ , \text{ i.e.,} \\ \|D_t\| &= \frac{1}{\sqrt{T}}o(1) \ . \end{aligned}$$

Let us now take polynomial $o(1)$ of type d/T^{ε} where d and ε are positive

constants to get

$$\begin{aligned}\|D_t\| &\leq \frac{1}{\sqrt{T}}\frac{d}{T^{\varepsilon}} \\ &= \frac{d}{T^{0.5+\varepsilon}} \ .\end{aligned}$$

Therefore, in this case

$$\begin{aligned}s(T) &\leq T\left(\frac{d}{T^{0.5+\varepsilon}}\right)^2 \\ &= \frac{d^2}{T^{2(0.5+\varepsilon)-1}} \\ &= \frac{d^2}{T^{2\varepsilon}} \ .\end{aligned}$$

Substituting this $s(T)$ into (5.13) we get the following corollary:

Corollary 3 *Under the conditions of Theorem 6, let T be known in advance. For every positive d, d_1, and ε, if $\|D_1\| \leq d_1$ and, for $t = 2, \ldots, T$,*

$$\|D_t\| \leq \frac{d}{T^{0.5+\varepsilon}} \ ,$$

then

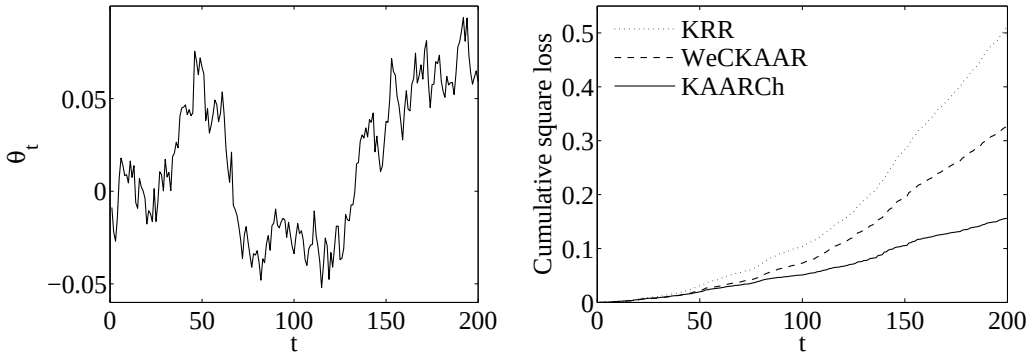
$$\begin{aligned}\mathrm{Loss}_T(KAARCh) &\leq \mathrm{Loss}_T(D) + O\left(T^{\max(0.5,(1-\varepsilon))}\right) \\ &= \mathrm{Loss}_T(D) + o(T) \ .\end{aligned}$$

□

Secondly, one may assume that there are only a few nonzero D_t, for $t = 2, \ldots, T$. In this case, the nonzero D_t can have greater flexibility.

5.4 Empirical Results

In this section we measure the empirical performance of our methods on an artificial dataset and on options implied volatility data.



(a) (b)

Figure 5.1: The behaviour of θ_t with time (a), approximating Brownian motion, and the cumulative loss suffered by KRR, WeCKAAR and KAARCh on the artificial dataset (b).

5.4.1 Artificial Dataset

Let $w_1, \ldots, w_T \in \mathbb{R}$ be T normally distributed random variables with mean 0 and variance σ^2, and $\theta_t = \sum_{i=1}^{t} w_i$. Drawing $\mathbf{x}_t \in \mathbb{R}$ from the interval $[0, 1]$ using a uniform distribution, we generate a dataset by the equation

$$\begin{aligned} y_t &= \theta_t' \mathbf{x}_t \\ &= \left(\sum_{i=1}^{t} w_i \right)' \mathbf{x}_t \ . \end{aligned}$$

The typical behaviour of a resulting θ_t with time can be seen in Figure 5.1 (a). In the normal regression setting (where the dependency does not change with time) this graph would simply be a flat line.

In our experiments, we set $T = 200$ and $\sigma = 0.01$, and repeated the procedure 20 times on randomly generated datasets. We used the linear kernel, which does not take any parameters. For WeCKAAR's $d_1, \ldots, d_T$ and KAARCh's $t_1, \ldots, t_T$, we used the numbers $1, \ldots, T$ to indicate the passage of time (these weights where subsequently normalised by dividing the values currently being used by their mean). To find good values of the parameter a, we simply found a value for which a method performed well on all the previously

seen examples. For the first step we took $a = 1$ since we have no history on which to perform validation. In Figure 5.1 (b) we show the mean over all runs of the cumulative square loss suffered by KRR, WeCKAAR and KAARCh on these datasets.

5.4.2 Options Implied Volatility Data

The Russian Trading System Stock Exchange (RTSSE) has provided us with data containing the details of options transactions on several underlying assets. Options are a type of derivative security (see Hull [2005] for more detailed information on options). They give the right to sell (put option) or buy (call option) an asset (like shares) which has current price S_t at some particular strike price K at a specific point in time in the future (at maturity). On a stock market, derivative securities are mainly used for hedging, that is, as an insurance against possible changes in the value of the underlying asset.

The accurate pricing of these options is an important problem. The most popular approach to pricing options is based on the Black-Scholes theory. This assumes that S_t follows an exponential Wiener process with constant volatility σ. The parameter σ cannot be observed directly, but it can be estimated from market data such as the price history (this estimate is called historical volatility). There are different types of options; we are interested in the so called European options. In this case the price at time t of call and put options, which we will denote by c_t and p_t respectively, are calculated by

$$\begin{aligned} c_t &= S_t N(d_1) - K N(d_2) \ , \text{ and} \\ p_t &= c_t + K - S_t \ , \end{aligned}$$

where $N(x)$ is the probability density function of the normal distribution with mean 0 and standard deviation 1, and given T which is the time until maturity in years,

$$\begin{aligned} d_1 &= \frac{\ln(S_t/K) + (\sigma^2/2)T}{\sigma\sqrt{T}} \ , \text{ and} \\ d_2 &= d_1 - \sigma\sqrt{T} \ . \end{aligned}$$

In practise this model is often violated. Given the current prices of options and the underlying asset we can find σ that satisfies the formulae above. This σ is known as the implied volatility and it exhibits a dependence on the strike price and the time to maturity. The curve showing the dependence of the implied volatility on the strike price is often called the volatility smile (see Hull [2005, Chapter 16]). If time to maturity is taken into account too, we get a volatility surface. The existence of volatility smiles and surfaces contradicts the Black-Scholes model. There is no generally recognised theory describing the phenomenon of implied volatility; however, it remains a useful parameter and traders at a stock exchange often use it to quote option prices.

We are interested in using learning theory methods for predicting implied volatilities without assuming any model for its behaviour. In our experiments we treat the implied volatility of a transaction as the outcome and the parameters of the transaction, namely the asset price, the strike price, the time to maturity, and whether an option is a put or a call, as the signal. In addition, we used the time at which transactions occurred as our methods' weights (see below).

Experimentation Methodology

We tested the performance of KRR, WeCKAAR, and KAARCh on options implied volatility data. As usual, we need to find good values for any tunable parameters of the methods employed. For the weights required by our new methods, specifically, WeCKAAR's $d_1, \ldots, d_T$ and KAARCh's $t_1, \ldots, t_T$, we use a real number representing the (normalised) time at which the transactions occurred. This number was provided by the RTSSE and in our experiments we subtract the minimum and add 1 so that time starts from 1. To normalise time we simply divide the values currently being used by their mean. The kernels used in our experiments are spline, polynomial degree 2, and RBF with $\sigma = 1$ (see Section 3.2.1)[2].

What remains to be done is to find good values for the parameter a (see (3.8) on page 40, (5.3), and (5.7)). This is achieved by applying a sliding

[2]These particular kernel parameters were chosen because they performed well in preliminary experiments.

window technique (the window size was set to 50), finding a good value for a on the first window and then making predictions on the next window in online mode. This is repeated for the whole dataset, meaning that the parameter a is updated after every 50 transactions. We applied normalisation schemes similar to what is described in Section 4.5.2. WeCKAAR and KAARCh function better if the mean of the outcomes is 0. We handle this by shifting the outcomes down by the mean of the outcomes of the previous window and later shift the predictions up by this same amount. Due to computational limitations, we ran experiments on 100 randomly selected segments containing 200 transactions from every dataset.

Results

In Table 5.1 we give the results obtained on options data for EERU, GAZP and RTSI. EERU are options on futures on shares of Unified Power Systems of Russia, GAZP are options on futures on shares of Gazprom, and RTSI are options on an RTSSE index (the numbers appended to the option names in the table specify different transaction periods). These options were chosen because they are popular and therefore are traded all the time.

The left half of the table shows the mean square losses suffered by WeCKAAR, KAARCh and KRR using different kernels, and also those of the proprietary method used at the RTSSE for comparison. The method used at the RTSSE is based on Kalman Filters (see Maybeck [1979, Chapter 1] for an introduction to Kalman Filters) and some assumptions on the shape of the volatility curve are made. This contrasts with our methods, where we make no assumptions at all (our methods' applicability is not limited to options data) and use general purpose kernels.

To measure the statistical significance of the difference between the results of our methods and those of the RTSSE we used the Wilcoxon signed rank test (see, for example, Hollander and Wolfe [1973]). When there is no statistical significance in the difference (we use the conventional 5% threshold) the corresponding p-value reported in the other half of the table is prefixed with an asterisk ($*$). A description of statistical significance and p-values is given in Section 4.5.3.

Table 5.1: Mean square losses suffered on options implied volatility data. All losses reported are $\times 10^3$, apart from those for EERU1206 which are $\times 10^2$.

Dataset: RTSI1206 (10126 transactions)						
	RTSSE: 2.91			Statistical significance of difference		
Kernel	KRR	WeCKAAR	KAARCh	KRR	WeCKAAR	KAARCh
Poly	36.56	2.19	**2.16**	4×10^{-4}	5×10^{-2}	$*\ 8\times10^{-2}$
Spline	2.63	**2.23**	2.24	9×10^{-4}	$*\ 9\times10^{-2}$	$*\ 9\times10^{-2}$
RBF	3.31	2.33	**2.31**	1×10^{-4}	5×10^{-2}	5×10^{-2}
Dataset: RTSI0307 (8410 transactions)						
	RTSSE: 2.78			Statistical significance of difference		
Kernel	KRR	WeCKAAR	KAARCh	KRR	WeCKAAR	KAARCh
Poly	8.29	2.40	**2.38**	7×10^{-12}	2×10^{-7}	2×10^{-7}
Spline	3.49	2.29	**2.29**	5×10^{-7}	2×10^{-6}	3×10^{-6}
RBF	3.87	2.33	**2.32**	3×10^{-7}	9×10^{-7}	1×10^{-6}
Dataset: GAZP1206 (9382 transactions)						
	RTSSE: 1.29			Statistical significance of difference		
Kernel	KRR	WeCKAAR	KAARCh	KRR	WeCKAAR	KAARCh
Poly	1.59	1.54	**1.53**	2×10^{-8}	1×10^{-8}	8×10^{-9}
Spline	5.21	**1.49**	1.49	4×10^{-10}	4×10^{-8}	4×10^{-8}
RBF	1.59	**1.47**	1.48	4×10^{-8}	6×10^{-8}	3×10^{-8}
Dataset: GAZP0307 (10985 transactions)						
	RTSSE: 2.13			Statistical significance of difference		
Kernel	KRR	WeCKAAR	KAARCh	KRR	WeCKAAR	KAARCh
Poly	3.16	2.45	**2.45**	6×10^{-11}	2×10^{-8}	2×10^{-8}
Spline	2.85	2.47	**2.47**	5×10^{-7}	4×10^{-8}	4×10^{-8}
RBF	3.53	**2.49**	2.49	1×10^{-7}	6×10^{-8}	5×10^{-8}
Dataset: EERU1206 (13152 transactions)						
	RTSSE: 1.47			Statistical significance of difference		
Kernel	KRR	WeCKAAR	KAARCh	KRR	WeCKAAR	KAARCh
Poly	162.43	**1.71**	1.72	3×10^{-7}	2×10^{-8}	4×10^{-8}
Spline	1.92	**1.65**	1.66	3×10^{-7}	4×10^{-7}	5×10^{-7}
RBF	6.36	**1.65**	1.65	3×10^{-7}	1×10^{-7}	2×10^{-7}
Dataset: EERU0307 (14776 transactions)						
	RTSSE: 4.74			Statistical significance of difference		
Kernel	KRR	WeCKAAR	KAARCh	KRR	WeCKAAR	KAARCh
Poly	5.49	4.58	**4.52**	2×10^{-4}	3×10^{-2}	4×10^{-2}
Spline	5.07	**4.49**	4.50	4×10^{-3}	4×10^{-2}	4×10^{-2}
RBF	5.83	**4.46**	4.49	5×10^{-4}	$*\ 6\times10^{-2}$	5×10^{-2}

5.5 Conclusion

In this chapter we introduced two new methods, WeCKAAR and KAARCh, to make predictions in online regression with changing dependencies. KAARCh has superior theoretical properties, including an upper bound on its loss that guarantees that at most it will suffer loss that is only a little more than that of any predictor that does not change very rapidly. Empirical experiments were carried out on artificial data and on six real world datasets on options implied volatility.

KAARCh's performance on the artificial dataset is much better than that of WeCKAAR and KRR. We attribute this to the fact that the artificial data was generated by a process that changes slowly at every step, which is ideal for KAARCh given its theoretical properties. The results achieved on the real world datasets by KAARCh and WeCKAAR are always better than those of KRR and very close to those of the RTSSE (and slightly better in half of them). The proprietary method used at the RTSSE was specifically designed for this application and is constantly monitored and tuned by experts to predict better. Therefore, it is remarkable that our methods perform comparably. These results show that our new methods KAARCh and (to a lesser extent) WeCKAAR are capable of handling changing dependencies and, in this context, are an improvement on standard regression techniques.

Chapter 6

Conclusion

In this dissertation we have considered the problem of regression, where we are given a signal and are required to output a real valued prediction. We suffer loss if our prediction does not perfectly match the outcome of the signal, which is given to us later. Early solutions to this problem include Least Squares (LS) and Ridge Regression (RR). LS gives a solution that fits the data too well, that is, it overfits the data. RR includes regularisation, in that it balances the goodness of the solution with its complexity, leading to better generalisation. RR can be seen as an improvement of LS, especially in the case where the data contains noise or outliers.

The Aggregating Algorithm (AA) is a technique that allows us to merge large pools of experts to make optimal predictions. These optimal predictions are almost as good as those of the best expert in the pool. Recently, the AA was applied to the problem of regression, to get the Aggregating Algorithm for Regression (AAR) and its nonlinear version, Kernel AAR (KAAR). Their main focus is on the online mode of learning, where signals and outcomes appear in a sequence and we are required to make a prediction at every step. Unlike Ridge Regression, AAR and KAAR have theoretical worst case upper bounds on their cumulative loss and they do not make any significant assumptions on the data.

6.1 Achievements

Our work is based on the application of the Aggregating Algorithm to problems in regression. We first analysed and improved the existing solutions and then derived a new algorithm for a harder regression problem.

6.1.1 Improving the Aggregating Algorithm for Regression

In empirical experiments we carried out (see Section 4.3 and Section 4.5), it is evident that most of the time KAAR does not perform as well as Kernel RR (KRR), the nonlinear version of Ridge Regression. KAAR is better than KRR only when the data is corrupted with lots of noise or contains severe outliers. Through analyses we found that this happens because KAAR includes some extra regularisation compared to KRR. Subsequently, we introduced new methods that through the use of an extra parameter can control or remove this extra regularisation. We have named our two main methods Iterated KAAR (IKAAR) and Controlled KAAR (CKAAR).

These methods are a generalisation of both KAAR and KRR as they can be made to behave as either one by choosing specific values for their extra parameter. In a comparison with KRR, we found that in a Bayesian interpretation, KAAR and our methods push KRR's predictions towards the mean of the outcomes by an amount proportional to the prediction's variance. Empirical experiments suggest that, in general, our new methods perform as well as or better than KRR and KAAR.

6.1.2 Regression with Changing Dependencies

It is usually assumed that the relationship between a signal and its outcome are fixed. However, there are cases where this is not true and we need to handle the possibility that this dependency changes with time. An example of this is predicting options implied volatility as explained in Section 5.4.2.

Our first solution to this problem was to modify our own method CKAAR, mentioned above, such that it gives less importance to older examples.

This was done since the relationship between older signals and their outcomes is outdated and therefore should be ignored to some degree. We called the resulting algorithm Weighted CKAAR (WeCKAAR).

The second solution that we present is an application of the Aggregating Algorithm (AA) to the pool containing all linear predictors that can change with time. The nonlinear version of the resulting method is called the Kernel Aggregating Algorithm for Regression with Changing dependencies (KAARCh). Given the AA's properties, we were able to derive an upper bound on KAARCh's cumulative loss. This states that at worst KAARCh will suffer a cumulative loss that is less or equal to that of any changing (nonlinear) predictor plus a small term, if the predictor does not change very rapidly.

Empirical results show that WeCKAAR and KAARCh are able to predict changing dependencies, consistently performing better than Kernel Ridge Regression (KRR). This is evident when running experiments on both artificial data and on options implied volatility data. On an artificial dataset, KAARCh performs much better than WeCKAAR and KRR. On the options implied volatility data, which was given to us by the Russian Trading System Stock Exchange (RTSSE), both our methods perform comparably to the proprietary technique currently being used at the RTSSE, even though they are applicable to a much wider class of problems.

6.2 Future Work

1. An interesting experiment would be to try out KRR, KAAR, IKAAR and CKAAR on an artificial dataset with different levels of noise and/or some severe outliers to see when our methods start to make an improvement. A possible dataset could be the Mexican Hat dataset generated by $y = \sin(|x|)/|x| + \varepsilon$ where $x \in [-10, 10]$ and ε is some noise (this dataset is also mentioned in Section B.1.1).

2. Our methods that improve the predictive performance of KAAR, like IKAAR and CKAAR, require one extra parameter. This parameter controls the amount of extra regularisation (compared to Ridge Regression)

used during prediction. There may be a correlation between the amount of noise or number of outliers in the data and the magnitude of this parameter. If this conjecture is correct, then it can be leveraged in at least two ways. First, it would be natural to try to automatically find a good value of this extra parameter by using some heuristics from the data. Secondly, going the other direction, one could use good values of this parameter, found through validation or otherwise, as a measure of the 'difficulty' or complexity of the data.

3. In a related note to the previous item, it would be interesting to see the effect of having the control parameter adapt on the fly. This could be achieved by making its magnitude proportional to the complexity of the signal for which a prediction is to be made. One straightforward way of measuring the complexity of a signal is by taking its norm.

4. To obtain KAARCh, we apply the AA to the pool consisting of all changing predictors. To do this, we pad our signals with zeros, to get a new vector of dimensionality nT, where n is the original dimensionality of the signal and T is the current step in time. This results in a matrix in the primal form of size $nT \times nT$. In AAR, where the experts do not change, the corresponding matrix is of size $n \times n$. It would be worthwhile to investigate whether it is possible to merge all changing predictors in a way that does not result in such a big matrix. In the dual form we do not have this problem, since the matrix is of size $T \times T$ which is the same as that of KAAR.

5. In this dissertation we derived an upper bound on the cumulative loss suffered by KAARCh. To measure the tightness of this bound it is necessary to derive a lower bound on KAARCh's loss, similar to what Vovk did for AAR in Vovk [2001, Section 3.3]. In Corollary 3 the term $\|D_t\| \leq d/T^{0.5+\varepsilon}$ tells us that KAARCh is competitive with experts that fluctuate slightly less than Brownian motion. If $\varepsilon = 0$ then the experts become more or less equivalent to Brownian motion and KAARCh cannot compete against them in a reasonable sense any more. This may be a possible direction to find a lower bound on KAARCh's loss.

6. In Section 5.3.3 we analyse KAARCh's loss upper bound and derive a new upper bound by restricting the experts we compete against. More analysis of KAARCh's upper bound, like considering other restrictions on the experts, can give us new bounds and a better insight into our new method.

7. More empirical experiments on KAARCh and WeCKAAR, especially applying them to other problems apart from predicting options implied volatility, will enable us to measure their predictive performance better. In addition, it would be very interesting to have an empirical comparison of our methods with similar techniques such as those described in Section 5.1.

8. To compute a prediction by KAARCh and WeCKAAR a $T \times T$ matrix has to be inverted. In the online mode of learning T may continue to grow indefinitely, which is not desirable. Our practical implementation of these methods is to drop older examples and work with a fixed window of examples instead. From a conceptual point of view this makes sense because older examples are given much less weight than newer ones and their omission should only make a small difference to the prediction, if any. It is desirable, therefore, to measure the difference between this approximation and the true prediction and verify that it is small and insignificant. A straightforward way of doing this is by running two separate empirical experiments on the same data: one computing the true predictions and the other just the approximations. The difference can then be calculated and advantages of speed of computation against loss of precision be measured. Another, more precise way of achieving this is to analyse mathematically the difference in the prediction formulae.

9. In computing predictions for WeCKAAR and KAARCh (and also KRR) in Section 5.4.2 we fix a window and use this as the history from which we get statistics and to find good values for any parameters required. This is somewhat crude as the parameter gets updated only at fixed intervals and it may be that better results can be achieved if the size of this window can change. Therefore, a better experimentation method

for KAARCh and WeCKAAR in the online mode of learning may give better predictive performance.

6.3 Final Remarks

The objectives set for the dissertation were met. We have analysed an existing application of the Aggregating Algorithm (AA) to regression and suggested improvements. These improved methods perform well when compared to competing techniques. We also consider the regression problem where the dependency of outcomes on signals can change with time. We give two solutions: the first is based on one of the improvements mentioned above, and the other is an application of the AA to merge all the predictors that can change with time. We show that the latter solution performs almost as well as the best slowly changing predictor in terms of the square loss it suffers. In empirical experiments both these methods perform well.

Bibliography

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

A. Asuncion and D. J. Newman. UCI machine learning repository (`http://www.ics.uci.edu/~mlearn/mlrepository.html`), 2007.

E. F. Beckenbach and R. Bellman. *Inequalities*. Springer-Verlag, Berlin, Germany, 1961.

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

C. J. C. Burges and V. Vapnik. A new method for constructing artificial neural networks. Technical Report ONR contract N00014-94-c-0186, AT&T Bell Laboratories, 1995.

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2–3):143–167, 2007.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, UK, 2006.

J. B. Conway. *A Course in Functional Analysis*. Graduate Texts in Mathematics. Springer, USA, second edition, 2000.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods)*. Cambridge University Press, Cambridge, UK, 2000.

H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Proceedings of the 1996 Conference on Advances in Neural Information Processing Systems*, volume 9, pages 155–161. The MIT Press, 1997.

A. Gammerman, Y. Kalnishkan, and V. Vovk. On-line prediction with kernels and the complexity approximation principle. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 170–176. AUAI Press, 2004.

D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44:1906–1925, 1998.

R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. The MIT Press, Cambridge, Massachusetts, USA, 2002.

M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.

M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. John Wiley & Sons, New York, USA, 1973.

J. C. Hull. *Options, Futures and Other Derivatives*. Prentice-Hall, New Jersey, USA, 6th edition, 2005.

V. I. Istratescu. *Fixed Point Theory, An Introduction*. Springer, USA, 1981.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, USA, 1979.

T. Melluish, C. Saunders, I. Nouretdinov, and V. Vovk. Comparing the Bayes and typicalness frameworks. In *Machine Learning: EMCL 2001. Proceedings of the Twelfth European Conference on Machine Learning*, volume 2167 of *Lecture Notes in Computer Science*, pages 360–371, Heidelberg, Germany, 2001. Springer.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, USA, 1997.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, second edition, 1994.

J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. D'Alché-Buc, editors. *Evaluating Predictive Uncertainty, Visual Object Categorization and Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, Heidelberg, Germany, 2006. Springer.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.

B. Schölkopf and A. J. Smola. *Learning with Kernels — Support Vector Machines, Regularization, Optimization and Beyond*. The MIT Press, Cambridge, Massachusetts, USA, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2005.

M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in kernel methods — support vector learning*, pages 285–292. MIT Press, Cambridge, MA, USA, 1999.

V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69 (2):213–248, 2001.

V. Vovk. On-line regression competitive with reproducing kernel Hilbert spaces. Technical Report arXiv:cs.LG/0511058 (version 2), arXiv.org e-Print archive, 2006.

V. Vovk. Aggregating strategies. In M. Fulk and J. Case, editors, *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56:153–173, 1998.

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer, New York, USA, 2005.

Appendix A

Lemmas

Lemma 4 *Given a matrix* $\mathbf{A}$*, a scalar* a *and* $\mathbf{I}$ *identity matrices of the appropriate size,*

$$(\mathbf{A}\mathbf{A}' + a\mathbf{I})^{-1}\mathbf{A} = \mathbf{A}(\mathbf{A}'\mathbf{A} + a\mathbf{I})^{-1} \ . \qquad \square$$

Proof

$$\begin{aligned}(\mathbf{A}\mathbf{A}' + a\mathbf{I})^{-1}\mathbf{A} &= (\mathbf{A}\mathbf{A}' + a\mathbf{I})^{-1}\mathbf{A}(\mathbf{A}'\mathbf{A} + a\mathbf{I})(\mathbf{A}'\mathbf{A} + a\mathbf{I})^{-1} \\ &= (\mathbf{A}\mathbf{A}' + a\mathbf{I})^{-1}(\mathbf{A}\mathbf{A}'\mathbf{A} + a\mathbf{A})(\mathbf{A}'\mathbf{A} + a\mathbf{I})^{-1} \\ &= (\mathbf{A}\mathbf{A}' + a\mathbf{I})^{-1}(\mathbf{A}\mathbf{A}' + a\mathbf{I})\mathbf{A}(\mathbf{A}'\mathbf{A} + a\mathbf{I})^{-1} \\ &= \mathbf{A}(\mathbf{A}'\mathbf{A} + a\mathbf{I})^{-1} \end{aligned} \qquad \blacksquare$$

Lemma 5 (Banach Fixed Point Theorem; see, for example, Istratescu [1981, Chapter 7]) *Let* (X, d) *be a nonempty complete metric space. Let* $T : X \mapsto X$ *be a contraction mapping on* X*, i.e., there is a real number* $0 \leq q < 1$ *such that*

$$d(Tx, Ty) \leq qd(x, y)$$

for all $x, y \in X$*. Then the map* T *admits one and only one fixed point* $\dot{x} \in X$*, that is,* $T\dot{x} = \dot{x}$*.* $\square$

Lemma 6 (Matrix Inversion by Partitioning; see Press et al. [1994, Section 2.7]) *Suppose that we are given a matrix* $\mathbf{A}$ *of size* $n \times n$ *partitioned*

in the following way

$$\mathbf{A} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix} ,$$

where $\mathbf{P}$ *and* $\mathbf{S}$ *are square matrices of size* $p \times p$ *and* $s \times s$ *respectively* $(p+s=n)$*, and* $\mathbf{Q}$ *and* $\mathbf{R}$ *of size* $p \times s$ *and* $s \times p$ *respectively (not necessarily square). If its inverse is partitioned in the same manner*

$$\mathbf{A}^{-1} = \begin{bmatrix} \widetilde{\mathbf{P}} & \widetilde{\mathbf{Q}} \\ \widetilde{\mathbf{R}} & \widetilde{\mathbf{S}} \end{bmatrix} ,$$

then $\widetilde{\mathbf{P}}$*,* $\widetilde{\mathbf{Q}}$*,* $\widetilde{\mathbf{R}}$ *and* $\widetilde{\mathbf{S}}$ *which have the same sizes as* $\mathbf{P}$*,* $\mathbf{Q}$*,* $\mathbf{R}$ *and* $\mathbf{S}$ *respectively, can be calculated by the following formulae (provided all the inverses exist):*

$$\begin{aligned}
\widetilde{\mathbf{P}} &= \mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}; \\
\widetilde{\mathbf{Q}} &= -\mathbf{P}^{-1}\mathbf{Q}(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}; \\
\widetilde{\mathbf{R}} &= -(\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}\mathbf{R}\mathbf{P}^{-1}; \\
\widetilde{\mathbf{S}} &= (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1} \ .
\end{aligned}$$

□

Lemma 7 (Riesz Representation Theorem; see, for example, Conway [2000, Theorem 3.4]) *Let* $\mathcal{H}$ *be a Hilbert space. If* $L : \mathcal{H} \mapsto \mathbb{R}$ *is a bounded linear functional, then there is a unique vector* $h_0 \in \mathcal{H}$ *such that* $L(h) = \langle h, h_0 \rangle$ *for every* $h \in \mathcal{H}$*. Moreover,* $\|L\| = \|h_0\|$*.* □

Lemma 8 *Let* $Q(\theta) = \theta'\mathbf{A}\theta + \mathbf{b}'\theta + c$*, where* $\theta, \mathbf{b} \in \mathbb{R}^n$*,* c *is a scalar and* $\mathbf{A}$ *is a symmetric positive definite* $n \times n$ *matrix. Then*

$$\int_{\mathbb{R}^n} e^{-Q(\theta)} d\theta = e^{-Q_0} \frac{\pi^{n/2}}{\sqrt{\det \mathbf{A}}} ,$$

where $Q_0 = \min_{\theta \in \mathbb{R}^n} Q(\theta)$*.* □

Proof Let $\theta_0 \in \arg\min Q$. Take $\xi = \theta - \theta_0$ and $\widetilde{Q}(\xi) = Q(\xi + \theta_0)$. It is easy to see that the quadratic part of $\widetilde{Q}$ is $\xi'\mathbf{A}\xi$. Since $0 \in \arg\min \widetilde{Q}$, the form has no linear term. Indeed, in the vicinity of 0 the linear term dominates over the quadratic term; if $\widetilde{Q}$ has a non-zero linear term, it cannot have a minimum

at 0. Since $Q_0 = \min_{\xi \in \mathbb{R}^n} \widetilde{Q}(\xi)$, we can conclude that the constant term in $\widetilde{Q}$ is Q_0. Thus $\widetilde{Q}(\xi) = \xi' \mathbf{A} \xi + Q_0$.

It remains to show that $\int_{\mathbb{R}^n} e^{-\xi' \mathbf{A} \xi} d\xi = \pi^{n/2} / \sqrt{\det \mathbf{A}}$. This can be proved by considering a basis where $\mathbf{A}$ diagonalises (or see Beckenbach and Bellman [1961, Section 2.7, Theorem 3]). ■

Lemma 9 *Let*

$$F(\mathbf{A}, \mathbf{b}, \mathbf{x}) = \min_{\theta \in \mathbb{R}^n} (\theta' \mathbf{A} \theta + \mathbf{b}' \theta + \mathbf{x}' \theta) - \min_{\theta \in \mathbb{R}^n} (\theta' \mathbf{A} \theta + \mathbf{b}' \theta - \mathbf{x}' \theta) \ ,$$

where $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$ *and* $\mathbf{A}$ *is a symmetric positive definite* $n \times n$ *matrix. Then* $F(\mathbf{A}, \mathbf{b}, \mathbf{x}) = -\mathbf{b}' \mathbf{A}^{-1} \mathbf{x}$. □

Proof It can be shown by differentiation that the first minimum is achieved at $\theta_1 = -\frac{1}{2} \mathbf{A}^{-1} (\mathbf{b} + \mathbf{x})$ and the second minimum at $\theta_2 = -\frac{1}{2} \mathbf{A}^{-1} (\mathbf{b} - \mathbf{x})$. The substitution proves the Lemma. ■

Lemma 10 (Sylvester's Determinant Identity) *For every matrix* $\mathbf{M}$ *the equality* $\det(\mathbf{I} + \mathbf{M}'\mathbf{M}) = \det(\mathbf{I} + \mathbf{M}\mathbf{M}')$ *holds (where* $\mathbf{I}$ *are identity matrices of the correct size).* □

Proof Suppose that $\mathbf{M}$ is an $n \times m$ matrix. Thus $(\mathbf{I} + \mathbf{M}\mathbf{M}')$ and $(\mathbf{I} + \mathbf{M}'\mathbf{M})$ are $n \times n$ and $m \times m$ matrices respectively. Without loss of generality we may assume that $n \geq m$ (otherwise we swap $\mathbf{M}$ and $\mathbf{M}'$). Let the columns of $\mathbf{M}$ be m vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$.

We have $\mathbf{M}\mathbf{M}' = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i'$. Let us see how the operator $\mathbf{M}\mathbf{M}'$ acts on a vector $\mathbf{x} \in \mathbb{R}^n$. By associativity $\mathbf{x}_i \mathbf{x}_i' \mathbf{x} = (\mathbf{x}_i' \mathbf{x}) \mathbf{x}_i$, where $\mathbf{x}_i' \mathbf{x}$ is a scalar. Therefore, if U is the span of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$, then $\mathbf{M}\mathbf{M}'(\mathbb{R}^n) \subseteq U$. In a similar way, it follows that $(\mathbf{I} + \mathbf{M}\mathbf{M}')(U) \subseteq U$. On the other hand, if $\mathbf{x}$ is orthogonal to $\mathbf{x}_i$, then $\mathbf{x}_i \mathbf{x}_i' \mathbf{x} = (\mathbf{x}_i' \mathbf{x}) \mathbf{x}_i = 0$. Hence $\mathbf{M}\mathbf{M}'(U^\perp) = 0$, where $U^\perp$ is the orthogonal complement to U with respect to $\mathbb{R}^n$. Consequently, $(\mathbf{I} + \mathbf{M}\mathbf{M}')|_{U^\perp} = \mathbf{I}$ (by $\mathbf{B}|_V$ we denote the restriction of an operator $\mathbf{B}$ to a subspace V). Therefore $(\mathbf{I} + \mathbf{M}\mathbf{M}')(U^\perp) \subseteq U^\perp$.

One can see that both U and $U^\perp$ are invariant subspaces of $(\mathbf{I} + \mathbf{M}\mathbf{M}')$. If we choose bases in U and in $U^\perp$ and then concatenate them, we get a basis

of $\mathbb{R}^n$. In this basis the matrix of $(\mathbf{I}+\mathbf{M}\mathbf{M}')$ has the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} ,$$

where $\mathbf{A}$ is the matrix of $(\mathbf{I}+\mathbf{M}\mathbf{M}')|_U$. It remains to evaluate $\det(\mathbf{A})$.

First let us consider the case of linearly independent $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. They form a basis of U and we may use it to calculate the determinant of the operator $(\mathbf{I}+\mathbf{M}\mathbf{M}')|_U$. However,

$$(\mathbf{I}+\mathbf{M}\mathbf{M}')\mathbf{x}_i = \mathbf{x}_i + \sum_{j=1}^{m} (\mathbf{x}_j'\mathbf{x}_i)\mathbf{x}_j \ ,$$

and thus the matrix of the operator $(\mathbf{I}+\mathbf{M}\mathbf{M}')|_U$ in the basis $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ is $(\mathbf{I}+\mathbf{M}'\mathbf{M})$.

The case of linearly dependent $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ follows by continuity. Indeed, m vectors in an n-dimensional space with $n \geq m$ may be approximated by m independent vectors to any degree of precision and the determinant is a continuous function of the elements of a matrix. ∎

Appendix B

Additional Empirical Results

In Section 4.5 we gave a detailed analysis of the empirical performance of KRR, KAAR, IKAAR, CKAAR, KOKO, and KRRV on the Gaze dataset and on the Boston Housing dataset. In this appendix we present preliminary results obtained on the artificial Mexican Hat dataset and on Abalone, Auto-MPG, Auto-Price, Relative CPU Performance, Servo and Wisconsin Prognostic Breast Cancer datasets (all from Asuncion and Newman [2007]).

B.1 Results

The results reported here are for the batch mode only and do not include results for the control methods KOKO and KRRV. Note that for all datasets except the artificial Mexican Hat dataset we performed 100 runs on random permutations of the data. The experimentation procedure used to obtain these results is the same as that explained in Section 4.5, but with the following differences:

- Kernels are not normalised;
- The translation of outcomes is done by shifting down all outcomes by the mean of the outcomes in the whole dataset and not just that of those in the training set;
- The actual value of the parameter a used is calculated by multiplying the

Table B.1: Validation parameters used for experiments on the Mexican Hat datasets.

Parameter Name	Values
Polynomial degree	6
Spline	(no parameters)
$\tilde{a}$	0.1
IKAAR m	$1, 2, \ldots, 5$
CKAAR b	$0, 0.01, \ldots, 1$

specified parameter $\tilde{a}$ by the mean of the diagonal of the kernel matrix, i.e., the trace divided by the dimension of the kernel matrix.

B.1.1 The Mexican Hat Dataset

The artificial Mexican Hat dataset is generated by the function

$$y_t = \frac{\sin(|x_t|)}{|x_t|} + \varepsilon_t \ ,$$

where x_t ranges over the interval $[-10, 10]$ and $\varepsilon_t \in \mathbb{R}$ is some noise. Plotting this function gives a graph that somewhat resembles the cross section of a tradition Mexican hat, hence the name. In our experiments we took 100 signals from the interval (and their corresponding outcomes) starting from -10 and going up to 10 with a step of 0.2, skipping 0. Two separate experiments were performed using noise (ε_t) taken from normal distributions with mean 0 and standard deviations 0.2 and 0.5.

1000 random permutations where taken and each permutation of the dataset was split into 50, 30 and 20 examples for training, validation and testing respectively. All the combinations of the parameters shown in Table B.1 were used during the validation stage. For this dataset we used a spline kernel and a polynomial kernel of degree 6 only. The results are in Table B.2. Note that the testing outcomes are not corrupted by noise, therefore the losses reported are due to the model error only.

Table B.2: Batch mode results on 1000 random permutations of the Mexican Hat datasets with noise from $N(0, 0.2)$ and $N(0, 0.5)$.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Mexican Hat dataset with noise from $N(0, 0.2)$						
Poly	$\times 10^{-2}$	$\times 10^{-2}$	KRR	KAAR	IKAAR	CKAAR
KRR	9.47	2.41		2×10^{-26}	1×10^{-58}	8×10^{-78}
KAAR	9.38	2.47	2×10^{-26}		$*\ 9\times10^{-1}$	$*\ 1\times10^{-1}$
IKAAR	9.38	2.44	1×10^{-58}	$*\ 9\times10^{-1}$		1×10^{-14}
CKAAR	**9.37**	2.43	8×10^{-78}	$*\ 1\times10^{-1}$	1×10^{-14}	
Spline	$\times 10^{-2}$	$\times 10^{-2}$	KRR	KAAR	IKAAR	CKAAR
KRR	7.55	2.09		7×10^{-71}	5×10^{-4}	2×10^{-53}
KAAR	7.65	2.17	7×10^{-71}		1×10^{-101}	4×10^{-134}
IKAAR	7.55	2.11	5×10^{-4}	1×10^{-101}		6×10^{-160}
CKAAR	**7.53**	2.10	2×10^{-53}	4×10^{-134}	6×10^{-160}	
Mexican Hat dataset with noise from $N(0, 0.5)$						
Poly	$\times 10^{-2}$	$\times 10^{-2}$	KRR	KAAR	IKAAR	CKAAR
KRR	11.36	3.43		9×10^{-109}	3×10^{-115}	4×10^{-125}
KAAR	**11.04**	3.35	9×10^{-109}		2×10^{-11}	4×10^{-3}
IKAAR	11.04	3.34	3×10^{-115}	2×10^{-11}		8×10^{-13}
CKAAR	11.05	3.34	4×10^{-125}	4×10^{-3}	8×10^{-13}	
Spline	$\times 10^{-2}$	$\times 10^{-2}$	KRR	KAAR	IKAAR	CKAAR
KRR	9.18	2.92		7×10^{-3}	3×10^{-4}	5×10^{-25}
KAAR	9.16	2.97	7×10^{-3}		2×10^{-32}	2×10^{-47}
IKAAR	9.13	2.96	3×10^{-4}	2×10^{-32}		2×10^{-59}
CKAAR	**9.12**	2.95	5×10^{-25}	2×10^{-47}	2×10^{-59}	

Table B.3: Validation parameters used for experiments on the Abalone dataset.

Parameter Name	Values
Polynomial degree	$2, 4, 6, 8$
Spline	(no parameters)
ANOVA spline order	$2, 4, 6, 8$
RBF σ	$2^{-10}, 2^{-8}, \ldots, 2^{2}$
$\tilde{a}$	$2^{-16}, 2^{-14}, \ldots, 2^{-4}$
IKAAR m	$1, 21, \ldots, 201$
CKAAR b	$0, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95, 0.99, 1$

B.1.2 The Abalone Dataset

The age in years of an abalone is determined by counting the number of rings in a cross section of its shell through a microscope and adding 1.5. The goal in the Abalone dataset is to predict the ages of abalones from 8 features corresponding to physical measurements. These measurements, which include the length and weight, are relatively easy to obtain.

The dataset contains 4177 examples and 100 random permutations where taken. Each permutation of the dataset was split into 1000, 100 and 3077 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table B.3 were used during the validation stage. The results are in Table B.4.

B.1.3 The Auto-MPG Dataset

The Auto-MPG dataset contains details of cars and their performance in terms of their fuel consumption in miles per gallon (mpg). In our experiment the mpg of a car was predicted given its features. 7 attributes were used, including features like the number of cylinders in the car's engine and its weight. Signals that have missing values were not included in the experiment.

The dataset used contains 392 examples and 100 random permutations where taken. Each permutation of the dataset was split into 200, 50 and 142 examples for training, validation and testing respectively. All the combinations

Table B.4: Batch mode results on 100 random permutations of the Abalone dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Poly			KRR	KAAR	IKAAR	CKAAR
KRR	2×10^4	2×10^5		$8{\times}10^{-20}$	$3{\times}10^{-21}$	$8{\times}10^{-7}$
KAAR	**4.82**	0.33	$8{\times}10^{-20}$		$9{\times}10^{-8}$	$1{\times}10^{-4}$
IKAAR	4.88	0.45	$3{\times}10^{-21}$	$9{\times}10^{-8}$		$4{\times}10^{-2}$
CKAAR	2×10^2	1×10^3	$8{\times}10^{-7}$	$1{\times}10^{-4}$	$4{\times}10^{-2}$	
Spline			KRR	KAAR	IKAAR	CKAAR
KRR	5.33	1.80		$4{\times}10^{-6}$	$9{\times}10^{-9}$	$3{\times}10^{-2}$
KAAR	**4.87**	0.32	$4{\times}10^{-6}$		$*\ 2{\times}10^{-1}$	$3{\times}10^{-2}$
IKAAR	4.87	0.31	$9{\times}10^{-9}$	$*\ 2{\times}10^{-1}$		$3{\times}10^{-2}$
CKAAR	4.99	0.53	$3{\times}10^{-2}$	$3{\times}10^{-2}$	$3{\times}10^{-2}$	
ANOVA			KRR	KAAR	IKAAR	CKAAR
KRR	5.08	1.05		$5{\times}10^{-4}$	$4{\times}10^{-6}$	$*\ 5{\times}10^{-1}$
KAAR	4.81	0.34	$5{\times}10^{-4}$		$*\ 9{\times}10^{-1}$	$4{\times}10^{-2}$
IKAAR	**4.79**	0.35	$4{\times}10^{-6}$	$*\ 9{\times}10^{-1}$		$2{\times}10^{-2}$
CKAAR	4.99	0.98	$*\ 5{\times}10^{-1}$	$4{\times}10^{-2}$	$2{\times}10^{-2}$	
RBF			KRR	KAAR	IKAAR	CKAAR
KRR	5.73	3.78		$2{\times}10^{-9}$	$7{\times}10^{-6}$	$7{\times}10^{-3}$
KAAR	**4.72**	0.33	$2{\times}10^{-9}$		$4{\times}10^{-7}$	$9{\times}10^{-6}$
IKAAR	4.86	0.56	$7{\times}10^{-6}$	$4{\times}10^{-7}$		$*\ 2{\times}10^{-1}$
CKAAR	5.54	3.76	$7{\times}10^{-3}$	$9{\times}10^{-6}$	$*\ 2{\times}10^{-1}$	

Table B.5: Validation parameters used for experiments on the Auto-MPG dataset.

Parameter Name	Values
Polynomial degree	$2, 3, \ldots, 6$
Spline	(no parameters)
ANOVA spline order	$2, 3, \ldots, 7$
RBF σ	$2^{-10}, 2^{-8}, \ldots, 2^2$
$\tilde{a}$	$2^{-10}, 2^{-9}, \ldots, 2^{-5}$
IKAAR m	$1, 11, \ldots, 81$
CKAAR b	$0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1$

Table B.6: Batch mode results on 100 random permutations of the Auto-MPG dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Poly			KRR	KAAR	IKAAR	CKAAR
KRR	8.87	3.13		6×10^{-24}	$*\ 9\times10^{-1}$	$*\ 1\times10^{-1}$
KAAR	13.57	5.77	6×10^{-24}		3×10^{-25}	2×10^{-25}
IKAAR	**8.83**	2.64	$*\ 9\times10^{-1}$	3×10^{-25}		$*\ 1\times10^{-0}$
CKAAR	8.84	2.94	$*\ 1\times10^{-1}$	2×10^{-25}	$*\ 1\times10^{-0}$	
Spline			KRR	KAAR	IKAAR	CKAAR
KRR	**8.14**	1.85		1×10^{-28}	$*\ 9\times10^{-1}$	$*\ 7\times10^{-2}$
KAAR	13.51	8.63	1×10^{-28}		8×10^{-28}	1×10^{-28}
IKAAR	8.27	2.02	$*\ 9\times10^{-1}$	8×10^{-28}		$*\ 2\times10^{-1}$
CKAAR	8.17	1.69	$*\ 7\times10^{-2}$	1×10^{-28}	$*\ 2\times10^{-1}$	
ANOVA			KRR	KAAR	IKAAR	CKAAR
KRR	**8.03**	1.74		1×10^{-27}	$*\ 6\times10^{-1}$	$*\ 1\times10^{-1}$
KAAR	12.72	9.61	1×10^{-27}		3×10^{-26}	3×10^{-27}
IKAAR	8.05	1.62	$*\ 6\times10^{-1}$	3×10^{-26}		$*\ 8\times10^{-1}$
CKAAR	8.05	1.59	$*\ 1\times10^{-1}$	3×10^{-27}	$*\ 8\times10^{-1}$	
RBF			KRR	KAAR	IKAAR	CKAAR
KRR	8.38	2.41		7×10^{-25}	$*\ 2\times10^{-1}$	2×10^{-2}
KAAR	13.59	8.43	7×10^{-25}		1×10^{-24}	4×10^{-25}
IKAAR	**8.36**	2.17	$*\ 2\times10^{-1}$	1×10^{-24}		$*\ 8\times10^{-1}$
CKAAR	8.37	2.34	2×10^{-2}	4×10^{-25}	$*\ 8\times10^{-1}$	

per kernel of the parameters shown in Table B.5 were used during the validation stage. The results are in Table B.6.

B.1.4 The Auto-Price Dataset

The aim for the Auto-Price dataset is to predict the price of a car from 15 features which include characteristics like length, weight, number of doors, engine type and insurance risk rating. Those signals in the original dataset that had missing features and 10 nominal features were removed.

The dataset used contains 159 examples and 100 random permutations where taken. Each permutation of the dataset was split into 100, 30 and 29 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table B.7 were used during the validation

Table B.7: Validation parameters used for experiments on the Auto-Price dataset.

Parameter Name	Values
Polynomial degree	$2, 3, \ldots, 6$
Spline	(no parameters)
ANOVA spline order	$2, 4, 6, 8, 10, 12, 15$
RBF σ	$2^{-10}, 2^{-8}, \ldots, 2^{2}$
$\tilde{a}$	$2^{-10}, 2^{-9}, \ldots, 2^{-5}$
IKAAR m	$1, 11, \ldots, 101$
CKAAR b	$0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1$

stage. The results are in Table B.8.

B.1.5 The Relative CPU Performance Dataset

The Relative CPU Performance dataset concerns itself with the problem of predicting the relative performance of a CPU given 6 features which include the size of its cache memory and its cycles per second. Two nominal features were removed.

The dataset used contains 209 examples and 100 random permutations where taken. Each permutation of the dataset was split into 150, 34 and 25 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table B.9 were used during the validation stage. The results are in Table B.10.

B.1.6 The Servo Dataset

For the Servo dataset the problem is to predict the rise time of a servomechanism in terms of 4 features: two continuous gain settings and two discrete choices of mechanical linkages.

The dataset contains 167 examples and 100 random permutations where taken. Each permutation of the dataset was split into 100, 40 and 27 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table B.11 were used during the validation

Table B.8: Batch mode results on 100 random permutations of the Auto-Price dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Poly	$\times 10^6$	$\times 10^6$	KRR	KAAR	IKAAR	CKAAR
KRR	**7.48**	8.84		2×10^{-19}	6×10^{-3}	2×10^{-4}
KAAR	13.51	8.36	2×10^{-19}		1×10^{-17}	2×10^{-19}
IKAAR	7.87	6.27	6×10^{-3}	1×10^{-17}		$*\ 9\times10^{-1}$
CKAAR	8.05	9.23	2×10^{-4}	2×10^{-19}	$*\ 9\times10^{-1}$	
Spline	$\times 10^6$	$\times 10^6$	KRR	KAAR	IKAAR	CKAAR
KRR	11.18	18.47		1×10^{-17}	$*\ 1\times10^{-1}$	2×10^{-2}
KAAR	21.89	12.85	1×10^{-17}		3×10^{-29}	2×10^{-20}
IKAAR	**10.03**	8.30	$*\ 1\times10^{-1}$	3×10^{-29}		$*\ 9\times10^{-1}$
CKAAR	11.66	17.96	2×10^{-2}	2×10^{-20}	$*\ 9\times10^{-1}$	
ANOVA	$\times 10^6$	$\times 10^6$	KRR	KAAR	IKAAR	CKAAR
KRR	8.12	13.59		9×10^{-16}	1×10^{-3}	3×10^{-2}
KAAR	10.92	6.91	9×10^{-16}		3×10^{-14}	5×10^{-14}
IKAAR	**7.26**	6.10	1×10^{-3}	3×10^{-14}		$*\ 3\times10^{-1}$
CKAAR	8.81	13.97	3×10^{-2}	5×10^{-14}	$*\ 3\times10^{-1}$	
RBF	$\times 10^6$	$\times 10^6$	KRR	KAAR	IKAAR	CKAAR
KRR	**7.13**	4.21		6×10^{-11}	2×10^{-3}	$*\ 6\times10^{-2}$
KAAR	9.55	5.56	6×10^{-11}		2×10^{-8}	6×10^{-8}
IKAAR	7.75	5.63	2×10^{-3}	2×10^{-8}		$*\ 2\times10^{-1}$
CKAAR	7.63	5.06	$*\ 6\times10^{-2}$	6×10^{-8}	$*\ 2\times10^{-1}$	

Table B.9: Validation parameters used for experiments on the Relative CPU Performance dataset.

Parameter Name	Values
Polynomial degree	$2, 3, \ldots, 6$
Spline	(no parameters)
ANOVA spline order	$1, 2, \ldots, 6$
RBF σ	$2^{-10}, 2^{-8}, \ldots, 2^{2}$
$\tilde{a}$	$2^{-10}, 2^{-9}, \ldots, 2^{-5}$
IKAAR m	$1, 11, \ldots, 101$
CKAAR b	$0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1$

Table B.10: Batch mode results on 100 random permutations of the Relative CPU Performance dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Poly	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	14.99	47.06		6×10^{-5}	$*\ 3\times10^{-1}$	$*\ 5\times10^{-1}$
KAAR	13.60	18.58	6×10^{-5}		2×10^{-7}	3×10^{-6}
IKAAR	**9.10**	15.98	$*\ 3\times10^{-1}$	2×10^{-7}		$*\ 9\times10^{-1}$
CKAAR	15.00	46.75	$*\ 5\times10^{-1}$	3×10^{-6}	$*\ 9\times10^{-1}$	
Spline	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	**4.34**	6.05		1×10^{-12}	$*\ 3\times10^{-1}$	$*\ 4\times10^{-1}$
KAAR	15.05	20.08	1×10^{-12}		6×10^{-14}	5×10^{-14}
IKAAR	7.70	14.00	$*\ 3\times10^{-1}$	6×10^{-14}		$*\ 4\times10^{-1}$
CKAAR	6.00	11.74	$*\ 4\times10^{-1}$	5×10^{-14}	$*\ 4\times10^{-1}$	
ANOVA	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	**3.77**	5.00		5×10^{-10}	$*\ 6\times10^{-1}$	$*\ 9\times10^{-1}$
KAAR	11.08	15.86	5×10^{-10}		8×10^{-8}	1×10^{-10}
IKAAR	6.87	12.54	$*\ 6\times10^{-1}$	8×10^{-8}		$*\ 4\times10^{-1}$
CKAAR	5.47	10.05	$*\ 9\times10^{-1}$	1×10^{-10}	$*\ 4\times10^{-1}$	
RBF	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	**5.83**	9.40		1×10^{-10}	$*\ 7\times10^{-1}$	$*\ 7\times10^{-1}$
KAAR	9.99	13.99	1×10^{-10}		9×10^{-7}	3×10^{-11}
IKAAR	7.12	12.48	$*\ 7\times10^{-1}$	9×10^{-7}		$*\ 1\times10^{-1}$
CKAAR	6.57	11.15	$*\ 7\times10^{-1}$	3×10^{-11}	$*\ 1\times10^{-1}$	

Table B.11: Validation parameters used for experiments on the Servo dataset.

Parameter Name	Values
Polynomial degree	$2, 3, \ldots, 6$
Spline	(no parameters)
ANOVA spline order	$1, 2, 3, 4$
RBF σ	$2^{-10}, 2^{-8}, \ldots, 2^{2}$
$\tilde{a}$	$2^{-10}, 2^{-9}, \ldots, 2^{-5}$
IKAAR m	$1, 11, \ldots, 101$
CKAAR b	$0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1$

Table B.12: Batch mode results on 100 random permutations of the Servo dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Poly	$\times 10^{-1}$	$\times 10^{-1}$	KRR	KAAR	IKAAR	CKAAR
KRR	**5.30**	3.74		8×10^{-15}	$*\ 6\times10^{-1}$	$*\ 4\times10^{-1}$
KAAR	6.93	4.60	8×10^{-15}		2×10^{-13}	4×10^{-19}
IKAAR	5.45	3.85	$*\ 6\times10^{-1}$	2×10^{-13}		$*\ 3\times10^{-1}$
CKAAR	5.38	4.00	$*\ 4\times10^{-1}$	4×10^{-19}	$*\ 3\times10^{-1}$	
Spline	$\times 10^{-1}$	$\times 10^{-1}$	KRR	KAAR	IKAAR	CKAAR
KRR	4.47	3.42		1×10^{-18}	3×10^{-2}	$*\ 6\times10^{-2}$
KAAR	6.51	4.54	1×10^{-18}		1×10^{-17}	1×10^{-20}
IKAAR	4.52	3.44	3×10^{-2}	1×10^{-17}		3×10^{-2}
CKAAR	**4.45**	3.48	$*\ 6\times10^{-2}$	1×10^{-20}	3×10^{-2}	
ANOVA	$\times 10^{-1}$	$\times 10^{-1}$	KRR	KAAR	IKAAR	CKAAR
KRR	**4.43**	3.49		5×10^{-18}	$*\ 2\times10^{-1}$	$*\ 7\times10^{-1}$
KAAR	6.25	4.36	5×10^{-18}		2×10^{-16}	3×10^{-19}
IKAAR	4.52	3.50	$*\ 2\times10^{-1}$	2×10^{-16}		$*\ 4\times10^{-1}$
CKAAR	4.45	3.53	$*\ 7\times10^{-1}$	3×10^{-19}	$*\ 4\times10^{-1}$	
RBF	$\times 10^{-1}$	$\times 10^{-1}$	KRR	KAAR	IKAAR	CKAAR
KRR	5.07	4.10		7×10^{-14}	$*\ 9\times10^{-1}$	$*\ 1\times10^{0}$
KAAR	6.73	4.75	7×10^{-14}		7×10^{-13}	2×10^{-15}
IKAAR	5.14	4.10	$*\ 9\times10^{-1}$	7×10^{-13}		$*\ 1\times10^{-1}$
CKAAR	**5.05**	3.91	$*\ 1\times10^{0}$	2×10^{-15}	$*\ 1\times10^{-1}$	

stage. The results are in Table B.12.

B.1.7 The Wisconsin Prognostic Breast Cancer Dataset

In the Wisconsin Prognostic Breast Cancer dataset the problem is to predict the time for a patient to recur (or her disease free time). 32 features are given, including characteristics of the cell nuclei and the tumour size. Four examples that had missing values and 2 features were removed.

The dataset used contains 194 examples and 100 random permutations where taken. Each permutation of the dataset was split into 100, 50 and 44 examples for training, validation and testing respectively. All the combinations per kernel of the parameters shown in Table B.13 were used during the validation stage. The results are in Table B.14.

Table B.13: Validation parameters used for experiments on the Wisconsin Prognostic Breast Cancer dataset.

Parameter Name	Values
Polynomial degree	$2, 3, \ldots, 6$
Spline	(no parameters)
ANOVA spline order	$8, 16, 24, 32$
RBF σ	$2^{-10}, 2^{-8}, \ldots, 2^{2}$
$\tilde{a}$	$2^{-10}, 2^{-9}, \ldots, 2^{-5}$
IKAAR m	$1, 11, \ldots, 101$
CKAAR b	$0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99, 1$

Table B.14: Batch mode results on 100 random permutations of the Wisconsin Prognostic Breast Cancer dataset.

Method	**MSE**	**SD**	**Statistical Significance of Difference**			
Poly	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	1.53	0.91		5×10^{-19}	7×10^{-15}	1×10^{-14}
KAAR	**1.15**	0.18	5×10^{-19}		2×10^{-4}	3×10^{-4}
IKAAR	1.19	0.23	7×10^{-15}	2×10^{-4}		$*\ 9\times10^{-1}$
CKAAR	1.20	0.28	1×10^{-14}	3×10^{-4}	$*\ 9\times10^{-1}$	
Spline	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	13.62	44.78		3×10^{-13}	5×10^{-12}	1×10^{-10}
KAAR	**1.14**	0.19	3×10^{-13}		1×10^{-2}	1×10^{-3}
IKAAR	1.17	0.22	5×10^{-12}	1×10^{-2}		$*\ 4\times10^{-1}$
CKAAR	1.17	0.23	1×10^{-10}	1×10^{-3}	$*\ 4\times10^{-1}$	
ANOVA	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	2×10^5	2×10^6		4×10^{-13}	6×10^{-7}	4×10^{-9}
KAAR	**1.15**	0.19	4×10^{-13}		2×10^{-4}	5×10^{-5}
IKAAR	2×10^5	2×10^6	6×10^{-7}	2×10^{-4}		$*\ 5\times10^{-1}$
CKAAR	1.38	1.95	4×10^{-9}	5×10^{-5}	$*\ 5\times10^{-1}$	
RBF	$\times 10^3$	$\times 10^3$	KRR	KAAR	IKAAR	CKAAR
KRR	1.16	0.19		2×10^{-8}	4×10^{-3}	2×10^{-2}
KAAR	**1.11**	0.19	2×10^{-8}		2×10^{-6}	2×10^{-7}
IKAAR	1.14	0.20	4×10^{-3}	2×10^{-6}		$*\ 8\times10^{-1}$
CKAAR	1.14	0.19	2×10^{-2}	2×10^{-7}	$*\ 8\times10^{-1}$	